REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	5/16/95	Final Technical Report 9/30/93-4/30/95

4. TITLE AND SUBTITLE

IMAGE UNDERSTANDING RESEARCH IN COMPLEX ENVIRONMENTS

5. FUNDING NUMBERS

F49620-93-1-0620

6. AUTHOR(S)

R. Nevatia
G. Medioni
K. Price

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Southern California
Institute for Robotics & Intelligent Systems
Powell Hall 204
Los Angeles, CA 90089-0273

AFOSR-TR-95

0495

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Advanced Research Projects Agency
3701 No. Fairfax Drive, Arlington, VA 22203-1711

Air Force Office of Scientific Research
Bldg. 410, Bolling AFB, DC 20332-6448

nm

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release;
Distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

We have undertaken a broad range of research for developing image understanding techniques to infer 3-D shape descriptions of the scene from sensed data which may consist of monocular images, stereo pairs, motion sequences or range data, to recognize the objects in the scene, and to keep integrated temporal descriptions in dynamic scenes. This reports details recent work in generating 3-D descriptions from range data or intensity data, object recognition using perceptual grouping and pose estimation, and representing and recognizing landmarks for indoor navigation.

SUBJECT TERMS

Computer vision; object recognition; perceptual grouping; range image analysis; navigation; generic object recognition

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

Image Understanding Research in Complex Environments

Grant No F49620-93-1-0620
Final Technical Report
September 30, 1993 to April 30, 1995
DARPA Order A.O. 7515/00
Program Code 0E20
Contractor: University of Southern California
Start Date: 9/30/93
Principal Investigator: Ram Nevatia
(213) 740-6427
Program Manager: Abraham Waksman
(202) 767-5025
R. Nevatia , G. Medioni and K. Price (Editors)
Institute for Robotics and Intelligent Systems
School of Engineering
University of Southern California
Los Angeles, CA 90089-0273
May 1995

Sponsored by Advanced Research Projects Agency ARPA Order No. 7515/00
Monitored by AFOSR Under Grant No F49620-93-1-0620.

Table of Contents

Preface

This report summarizes research for Grant No F49620-93-1-0620 for the 19 months from September 1993 to April 1995. We give a general overview of the work in the introduction with more detail provided in later chapters.

1 Introduction

We have undertaken a broad range of research for developing image understanding techniques to infer 3-D shape descriptions of the scene from sensed data which may consist of monocular images, stereo pairs, motion sequences or range data (such as from LADAR), to recognize the objects in the scene, and to keep integrated temporal descriptions in dynamic scenes. These activities are central to almost any IU application task such as target recognition, photo-interpretation, navigation and object manipulation. Much of the past work in IU attempted to find solutions to these problems in highly specialized domains. While this approach may be necessary for some short-term applications, it requires extensive development for each new task. Worse yet, such techniques fail to be robust for the specific applications, as the limiting assumptions are easily violated. Instead, we follow an approach of *generic vision* that applies to large classes of objects and scenes and is based on broad and generic assumptions. We also use mathematically rigorous constraints derived from the geometry of the image formation process.

This chapter gives a brief outline of our work over the last 18 months with references to the later chapters of this Final Technical Report.

1.1 Analysis of Range Images

The goals of our effort in Range image understanding are to generate rich descriptions from sensed 3-D data. These descriptions should be segmented and capture both the volumetric and surface information related to objects. One of the applications is the automatic generation of 3-D models from multiple range images. We describe in more detail four specific aspects of our research:

- the integration of multiple range images into a surface representation of the object (in Chapter 2)
- a framework to handle complex objects with holes or multiple components (in Chapter 3)
- an approach to generate volumetric part descriptions from a surface representation of an object (in Chapter 4)
- the inference of surface shape from sparse three-dimensional data using perceptual organization (in Chapter 5).

1.2 Object Descriptions from Intensity Images

This is one of the most difficult, but important, tasks in IU. Scene segmentation is difficult, as different types of features such as object boundaries, surface orientation isocontinuities, surface markings, shadows and noise cannot be directly distin-

guished. We use a process of perceptual organization to compute the higher-level descriptions in such cases. To be generic, perceptual grouping methods must use general methods. One method propagates the influence of local features over large vector fields and finds the most salient features. For higher level groupings, we use methods based on utilizing projective properties of contours of a class of objects. We have chosen generalized cylinders (GCs) as suitable volumetric representations. A few types of GCs (and their combinations) can represent a large fraction of the man-made objects in our environment. In recent research, we have developed some very powerful invariant (and quasi-invariant) symmetry properties of projected contours of GCs. Our studies indicate that we can use these properties to segment objects, fill gaps even in presence of occlusion and infer 3-D shapes from monocular images. This is described in more detail in Chapter 6.

1.3 Object Recognition

Most object recognition systems today address the problem of finding the location and orientation of an exactly known rigid object in a scene.However, these approaches cannot be extended to more general scenarios because objects may be very similar while being geometrically different. Consider for instance two different airplanes which have similar features but different geometries. In other words, generic recognition should not make use of methods based purely on the exact geometric structure of the object. It is clear that the only way to solve this difficult problem is to reason about parts and their arrangements.We describe two aspects of our research designed to achieve this difficult goal:

- the generation of rich, stable descriptions from images, and the use of perceptual grouping laws to achieve this task (see Chapter 7)
- the development of an alignment-like pose estimation technique for multi-part curved objects (see Chapter 8).

1.4 Indoor Navigation and Dynamic Scene Analysis

We have been developing a vision system for indoor robot navigation. This system is based on a Denning mobile robot with a trinocular vision system. Our objective is to use generic descriptions of the path (go past the desk on your right and go through the first open door) rather than a detailed specific map. Currently our robot is able to navigate in laboratory environments avoiding obstacles and using objects such as doors and desks for landmarks.

We have also studied the problem of navigation in an environment which we model as we go. Specifically, we consider the problem of building a model of a scene, so that a similar sensor could, at a later time, orient itself with respect to this representation. We describe two aspects of this work in navigation:

- the representation of the spatial environment for indoor navigation (see Chapter 9)

- the recognition and location of generic objects (see Chapter 10).

2 Description of Complex Objects from Multiple Range Images Using an Inflating Balloon Model

Yang Chen and Gérard Medioni

We address the problem of constructing a complete surface model of an object using a set of registered range images. The construction of the surface description is carried out on the set of registered range images. Our approach is based on a dynamic balloon model represented by a triangulated mesh. The vertices in the mesh are linked to their neighboring vertices through springs to simulate the surface tension, and to keep the shell smooth. Unlike other dynamic models proposed by previous researchers, our balloon model is driven only by an applied inflation force towards the object surface from inside of the object, until the mesh elements reach the object surface. The system includes an adaptive local triangle mesh subdivision scheme that results in an evenly distributed mesh. Since our approach is not based on global minimization, it can handle complex, non-star-shaped objects without relying on a carefully selected initial state or encountering local minimum problem. It also allows us to adapt the mesh surface to changes in local surface shapes and to handle holes present in the input data through adjusting certain system parameters adaptively. We present results on simple as well as complex, non-star-shaped objects from real range images.

2.1 Introduction

The task of surface description using 3-D input can be described as finding a fit of a chosen representation (model surface) to the input data. This process can be formalized in a number of ways involving the minimization of a system functional that explicitly or implicitly represents the fit of the model to the input data. Another very important aspect of such a system is to construct a *mapping* or *correspondence* between the surface of an object and the structure of the model. This mapping exists because the surface of the model and the surface of the object are topologically equivalent, considering genus zero type of objects. Therefore there exists a one-to-one mapping between the model structures and the object surface elements. Previous researchers have studied such mappings in a variety of ways using different representation schemes and model fitting methods. Examples of these approaches include the dynamic system using energy minimization in [4] and the dynamic mesh in [12] and [13]. The drawbacks of these approaches is that they must rely on an initial guess of the model structure which is relatively close to the shape of the object. The reason is that, in the absence of mapping or correspondence information, some other approxi-

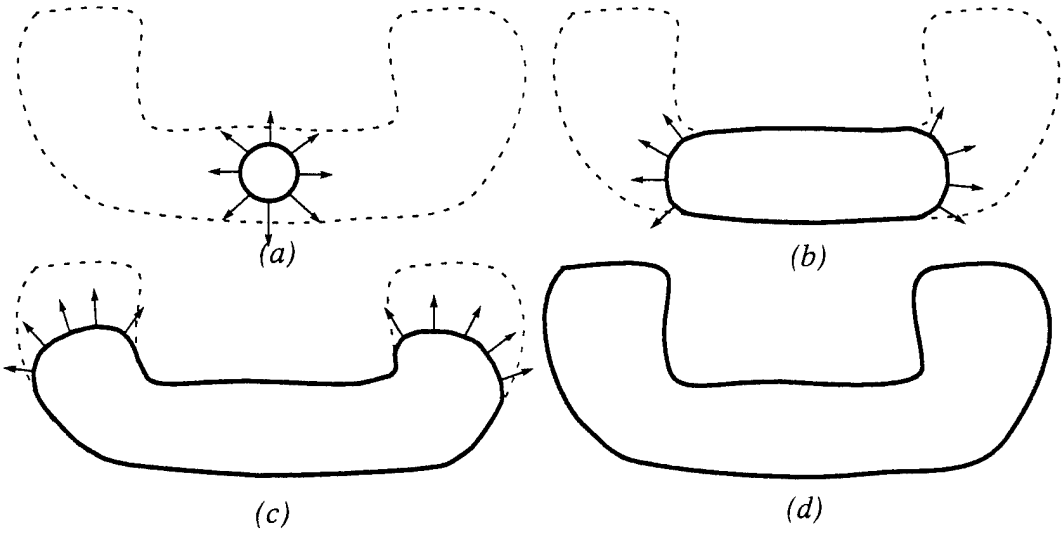


Figure 2.1 The inflating balloon model as illustrated in a 2-D case: (a) the initial state, (b) and (c) intermediate states, (d) the final state.

mations have to be used, such as the nearest data point to the model [13] in order for the system to converge to the desired results under the attraction force between the correspondence points in the model and the data. Such an approximation would have problems in cases where there are data points that are not the closest to their true corresponding points of the model, and thus inevitably lead the system towards a sub-optimal situation (local minimum). This is why those approaches can only deal with star-shaped objects.

In this paper we present a new approach for surface description using a dynamic balloon model represented by a triangular mesh. We start with a small triangulated shell placed inside the object and apply a uniform inflation force on all vertices in the direction normal to the shell's surface. The vertices are also linked to their neighboring vertices through springs to simulate the surface tension and to keep the shell smooth. The applied inflation force moves the vertices towards the object surface until they "land" on it. This process is similar to that of blowing up a balloon placed inside the hollow object until it fits the shell of the object. Thus the goal of mapping the model to the object surface is achieved through the physics of a growing balloon in a very natural way (see Figure 2.1). Of course, we also need to handle noisy data, and holes (lack of data), as described later.

Our system is not based on global minimization methods, and it can make decisions based on local information about the shape of the object surface. During the process of the growth of the triangular mesh, the triangles will be subdivided dynamically to reduce spring tension and to allow the mesh surface area to increase in order to cover the larger object surface. As the mesh expands and the vertices start to reach the object surface, the entire mesh surface is gradually subdivided into pieces

of connected triangular regions, which allows us to treat the surfaces in a local context by tailoring the parameters, and possibly strategy, of the system in dealing with each region separately based on local information.

Another aspect of our approach is its role in data integration. Most of the previous research make use of *scattered* 3-D points or a *single range image* as input. Very few researchers (e.g. [10]) try to use *multiple* densely sampled range images. There are two difficulties in using these range images. First the images must be precisely registered. We have previously presented a method [1] to register multiple range images, which is used to register the range images used in this paper. The second is the issue of integration. Integration can not be performed without an appropriate representation for the integrated data. While star-shaped objects can be simply mapped onto a unit sphere, and the integration can then be performed easily [1]. This is not true for complex objects, since it is difficult to find an integrated representation. Thus finding a suitable representation is very important. While it is not the main theme of this paper, we believe that our approach leads to a good solution to combine and take advantage of multiple range images in surface description for complex objects in terms of integration.

In the following sections, we first review some of the related previous work and then present our balloon model in detail. Section [2.3] describes our surface model and how it works, Sections [2.4] and [2.5] define the dynamics of the system. Section [2.6] explains the adaptive mesh subdivision scheme. In Sections [2.7] and 8, we give an algorithmic description of our system and describe how to set system parameters. Section Chapter 2 discusses issues on how to adapt the parameters locally and dealing with noise. Several test results from real range images, from both simple and complex objects, are presented in Section [2.10]. The conclusion follows in Section [2.11].

2.2 Related Work

Dynamic mesh models have been proposed by previous researchers for shape description. [12] introduced a shape reconstruction algorithm using a dynamic mesh that can dynamically adjust its parameters to adapt to the input data. [13] then extended this approach by introducing an attraction force from the 3-D input for shape description. [6] also proposed a similar system with dynamic nodal addition/deletion for shape description and nonrigid object tracking. [4] proposed a deformable model with both internal smoothness energy and external forces from both the input data and features. There exist other deformable model approaches that differ in the representation schemes of the model and in the approaches to solving the system [5][11].

The main difference between our method and those used by previous researchers is that we do not *explicitly* introduce a data force into our model, as discussed in details in the following sections. Our model is driven by an inflation force introduced inside the balloon. Balloon models have been used by [3] and [7], but in these ap-

proaches, the introduced balloon force is used mostly to overcome noise in the data so that the system can converge to the desired results more easily.

2.3 The Inflating Balloon Model

Our balloon model is represented by a shell of triangulated patches. The initial triangulated shell is an icosahedron. A triangulated shell can be either considered as a mesh consisting of triangular patches or a mesh consisting of vertices (or nodes) connected to their neighbors. In the following discussion, a mesh element may refer to either a vertex or a triangle patch. But in this paper, we mainly explore the properties of the vertices.

When placed inside the object, and under the influence of the inflation force, the shell grows in size as the vertices move along the mesh surface normal in the radial direction, maintaining an isotropic shape, until one or more vertices reaches the object surface. During the process of inflation, the triangles may be subdivided adaptively, which also creates new vertices. Once it reaches the surface, a vertex is considered *anchored* to the surface and thus can no longer move freely. The remaining vertices, under the influence of the anchored vertices, will gradually change their course of movement, until finally reaching their corresponding surface point. As can be seen from this process, the movement of the vertices is not influenced directly by any force from the surface of the object. This seems to be bad from the viewpoint of a fitting process, which tries to minimize some distance measure between the object surface and the model. But this is important to us because our main concern is to find the mapping between the mesh elements and the object surface. Not using any attraction force from the surface data allows us to avoid incorrect mappings, which is a similar situation to the local minimum problem in energy minimization approaches. An example of the growing balloon is shown in Figure 2.2 .

2.3.1 The Correspondence Problem

So far, we have not discussed how to test whether the mesh has reached the surface of the object. This is the key difference between our approach and other dynamic model systems or energy minimization systems. In order to test whether a vertex has reached the object surface, one must measure the distance between the mesh surface and some point on the object surface. Ideally this point on the object surface should be the corresponding point of the vertex, which is not possible before the vertex reaches the surface. Previous researchers have used the closest point on the object surface to an mesh element as an alternative, but it may provide incorrect information. In [7] the distance from the data points on the surface to the nearest model point is used instead, which is an improvement over the above approach. This approach, however, is not practical when there is a large number of surface sample points from the object, as in our case.

In our approach, we look for potential corresponding points only in the direction normal to the mesh surface. This is the best knowledge locally available to the points

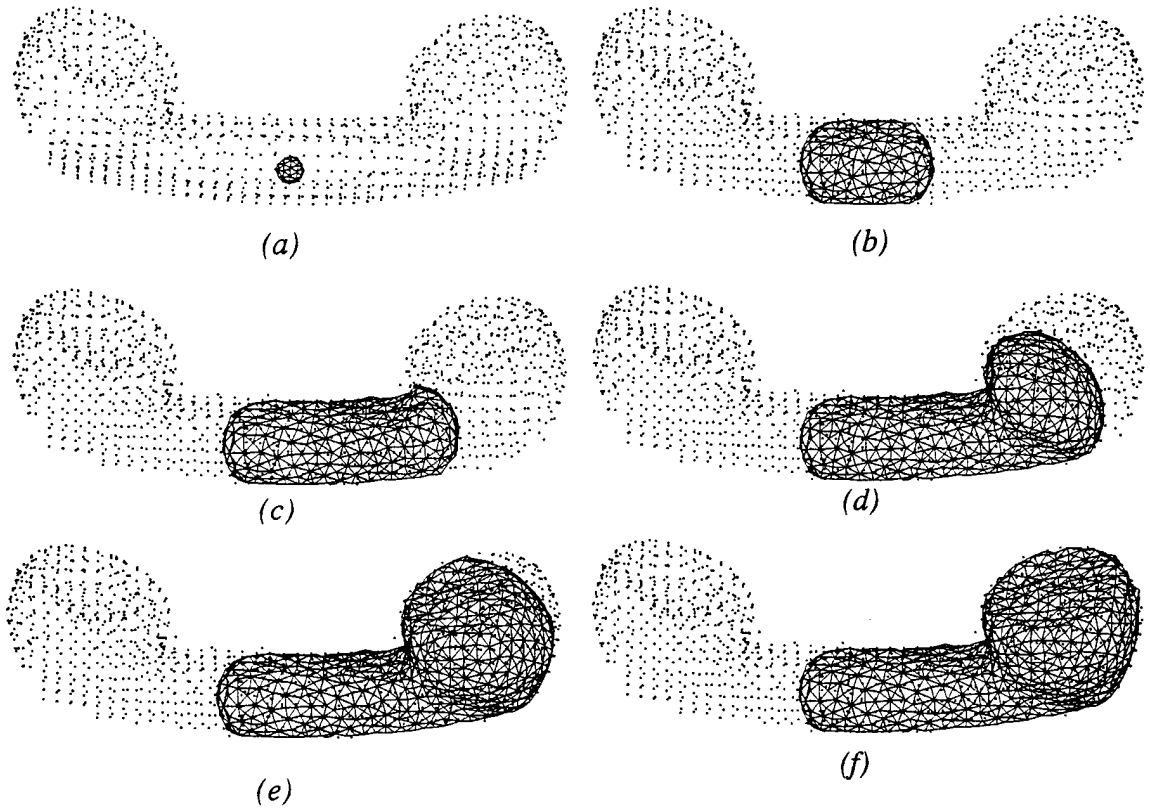


Figure 2.2 Stages of an inflating balloon inside the Phone, showing the movement of the right front only (see Section Chapter 2). The wireframes are superimposed with sample points from the used range images.

on the mesh surface at any time during the mesh's growing process, because the equivalent mesh surface movement in the neighborhood of a mesh element is only in the direction normal to the mesh surface. So it is only natural to look for corresponding point from the object surface in the direction of the normal, which also changes in the process of inflation. In our implementation, this is done by computing the prospective correspondence point P (Figure 2.3), the closest intersection of a line in the normal direction and the object surface represented in range images (see Appendix for details). Once the intersection is found, the distance from the mesh surface to the intersection can be used as a measure of whether the mesh has reached the object surface.

When there are holes in the input data (parts of the object surface not covered by the input data), we will not be able to find the intersections described above. In such cases, there are no prospective correspondence points for the affected vertices and thus there is no reason to continue applying inflation force. This kind of decision,

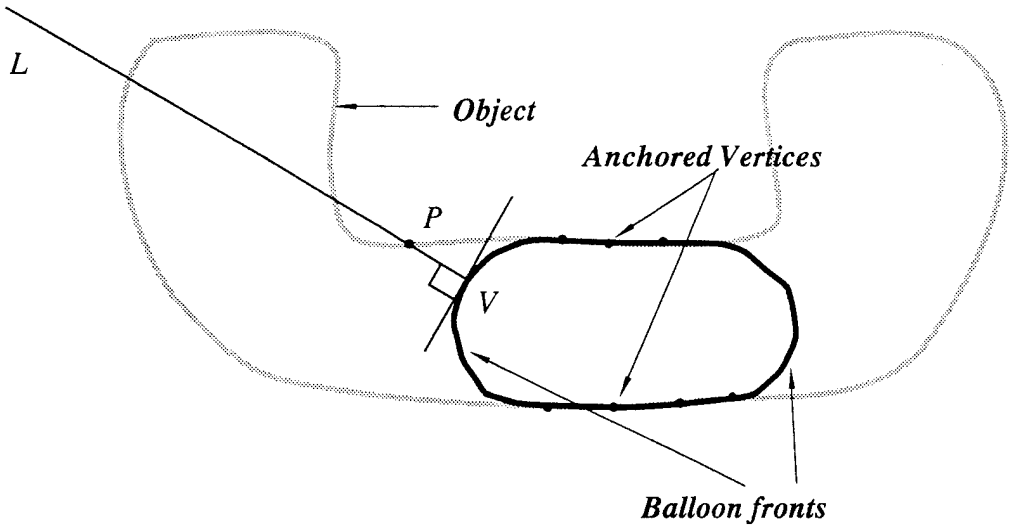


Figure 2.3 Line-Surface intersection: searching for a correspondence point in the normal direction.

however, should not be made locally for each vertex point. We will discuss the handling of holes when we discuss our algorithm in details later, in Section [2.8].

2.4 A Simplified Dynamic Model

The motion of any element i on the model surface can be described by the following motion equation [12]:

$$m_i\ddot{\boldsymbol{x}}_i + r_i\dot{\boldsymbol{x}} + \boldsymbol{g}_i = \boldsymbol{f}_i, \quad i = 1 \ldots N \tag{2.1}$$

where $\boldsymbol{x}_i$ is the location of the element, $\dot{\boldsymbol{x}}_i$ and $\ddot{\boldsymbol{x}}_i$ are the first and second derivatives with respect to time, m_i represents the mass, r_i is the damping coefficient, $\boldsymbol{g}_i$ is the sum of internal forces from neighboring elements due to, e.g., spring attachments and $\boldsymbol{f}_i$ is the external force exerted on the element. Because of the nonlinear nature of the forces $\boldsymbol{g}_i$ and $\boldsymbol{f}_i$ involved, the systems of ordinary differential equations in Equation (2.1) can be solved using explicit numerical integration [12].

The dynamic system will reach the equilibrium state when both $\dot{\boldsymbol{x}}_i$ and $\ddot{\boldsymbol{x}}_i$ become 0, which can take a very long time since it is usually an exponential process. A simplified system can be obtained if we make $m_i = 0$, and $r_i = 1$ for all i, in which case Equation (2.1) reduces to

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_i - \boldsymbol{g}_i, \quad i = 1 \ldots N \tag{2.2}$$

There are several reasons for this simplification. First, a zero-inertia system is simpler and easier to control. Second, since there is no inertia, the system will evolve faster in general. Although we are not seeking an equilibrium state for the entire sys-

tem, a simplified system will help speed up reaching local equilibrium states and therefore accelerate the overall dynamic process. Third, a simplified system involves less computation. Also, since we do not intend to have a special treatment for any particular elements in the mesh at this time, all r_i should be equal, in which case we can normalize the parameters so that $r_i = 1$. The set of first-order differential equations in Equation (2.2) has a very simple explicit integration form as follows:

$$\boldsymbol{x}^{t+\Delta t} = \left(\boldsymbol{f}_i^t - \boldsymbol{g}_i^t\right)\Delta t + \boldsymbol{x}^t \tag{2.3}$$

2.5 Spring Force and Inflation Force

The spring force exerted on vertex i by the spring linking vertex i and j can be expressed as [12]:

$$\boldsymbol{s}_{ij} = \frac{c_{ij}e_{ij}}{\|\boldsymbol{r}_{ij}\|}\boldsymbol{r}_{ij} \tag{2.4}$$

where c_{ij} is the stiffness of the spring, $e_{ij} = \|\boldsymbol{r}_{ij}\| - l_{ij}$ is the spring deformation, $\boldsymbol{r}_{ij} = \boldsymbol{x}_j - \boldsymbol{x}_i$, $\|\ \|$ is the vector length operator and l_{ij} is the natural length of the spring. The total spring force $\boldsymbol{g}_i$ a vertex receives is the vector sum of spring forces from all springs attached to it.

The inflation force a vertex receives takes the form of:

$$\boldsymbol{h}_i = k\hat{\boldsymbol{n}}_i \tag{2.5}$$

where k is the amplitude of the force and $\hat{\boldsymbol{n}}_i$ is the direction normal to the local model surface. In our implementation, the normal at a mesh vertex is estimated from the vector sum of the normal vectors of the surrounding triangles:

$$\hat{\boldsymbol{n}}_i = \frac{\boldsymbol{n}_i}{\|\boldsymbol{n}_i\|}, \qquad \boldsymbol{n}_i = \sum \frac{(\boldsymbol{n}_{ij} + \boldsymbol{n}'_{ij})}{\|(\boldsymbol{n}_{ij} + \boldsymbol{n}'_{ij})\|}. \tag{2.6}$$

where $\boldsymbol{n}_{ij}$ is the direction normal to the jth triangle $T_j \in \{T_j\}$ surrounding the vertex, and $\boldsymbol{n}'_{ij}$ is the direction normal to triangle T'_j that is the neighbor of T_j but $T'_j \notin \{T_j\}$. This estimation is more stable than the one we get when only the triangles in $\{T_i\}$ are used.

2.6 Subdivision and Adaptation of the Triangular Mesh

In a simulated physical system, during the process of the growth of the mesh model, the mesh triangles increase in size, and tensions due to the spring force also build up, which eventually stops the movement of the mesh, as the inflation force and the spring tension equalize. This is not desirable in our system since we do not consider force from the input data, so that an equilibrium state does not mean a good fit. In order to keep the balloon growing, we can keep the inflation force unchanged

(which actually means to keep on inflating) and at the same time reduce the spring tension by subdividing the triangles in the mesh into smaller triangles. Alternatively, we can increase the inflation force and allow the spring tension to increase. But increasing the inflation force also has the side effect of increasing the maximum displacement. As can be seen from Equation (2.3), the spring force g_i usually acts as a balance force to the inflation force $f_i = h_i$ (assuming a convex local structure), thus the maximum displacement is directly related to the inflation force once a time step is chosen. So we choose not to increase the inflation force in our system, but to subdivide the mesh instead.

The purpose of subdividing triangles is twofold. Once a triangle is subdivided, the sides of the triangles becomes shorter and if we keep the natural length and stiffness of the springs constant, the spring tension is reduced. Also, subdividing the triangles helps maintain an evenly distributed mesh. Subdividing triangles in a certain region, as will be discussed later, also allows the mesh surface to adapt to the local object surface geometry without affecting other parts of the mesh surface.

Before introducing the details of the triangle subdivision process, we first define some terms.

A vertex is said to be *anchored* if it has reached the object surface and has been marked as such. A triangle is said to be anchored if all of its vertices are anchored. At any time in the mesh growing process, the triangles in the mesh can be classified into anchored triangle regions, consisting of anchored triangles, and unanchored triangle regions, consisting of movable triangles, called *front*. Each front is a connected component of triangles, in which two triangles are said to be connected iff they share an edge.

2.6.1 Adaptive Triangle Mesh Subdivision

Triangle subdivision is carried out only on the front, since anchored triangles are not allowed to move. This allows the triangular mesh to adapt to the object surface better without globally adjusting the position of all vertices. A good subdivision scheme is one that yields an evenly distributed mesh and produces few degenerate (i.e. long and thin) triangles. The algorithm that we use in this paper first selects a set of triangles that needs to be subdivided through bisection. Then, after these triangles are bisected on their longest edges, adjacent triangles are also bisected or trisected to make the triangles *conforming*, the state in which a pair of neighboring triangles either meet at the a vertex or share an entire edge. In our implementation, only those triangles that exceed certain size limit are subdivided first. The algorithm presented below is adapted from Algorithm 2 (local) in [8], which is developed for refining triangular mesh for finite element analysis.

2.6.2 Algorithm 1

Let τ_{f0} be the set of the triangles from a given front, and $\tau_0 \subset \tau_{f0}$ is the selected set of triangles to be subdivided.

1) Bisect T by its longest edge, for each $T \in \tau_0$.
2) Find $R_1 \subset \tau_0$ the set of non-conforming triangles generated in step 1. Set $k \leftarrow 1$.
3) For each $T \in R_k$ with non-conforming point $P \in T$ (mid-point on the non-conforming edge): (a) bisect T by the longest edge; (b) if P is not on the longest edge of the T, then join P with the midpoint of the longest edge.
4) Let τ_0^k be the triangulation generated in step 3. Find $R_{k+1} \subset \tau_0^k$ the set of non-conforming triangles generated in step 3.
5) If $R_{k+1} = \{0\}$, stop, the subdivision is done. Else, set $k \leftarrow k + 1$ and go to step 3.

This subdivision algorithm has the feature that the subdivision is only propagated towards large triangles from the longest edge of a subdivided triangle. It is also proven that the resulting triangles' smallest inner angle is lower-bounded by half of the smallest inner angle of the original triangles [8].

This algorithm, however, does not guarantee that the triangles on the boundary areas of a front conform with the rest of the triangles in the triangulation. Hence, after the algorithm terminates, we must bisect the affected non-conforming triangles accordingly. Thus we have:

2.6.3 Algorithm 2

1) Carry out Algorithm 1 on the set of triangles $\tau_0 \subset \tau_{f0}$.
2) For each non-conforming triangle $T \notin \tau_{f0}$ but connected to τ_{f0}, bisect T by its non-conforming edge.

An example of the result from this algorithm is shown in Figure 2.4 , where triangle A is to be subdivided and C does not belong to the region (front). As can be seen from the figure, the subdivision is propagated to B, and finally C is bisected to make the triangles at the region boundary conforming (step [2] above).

2.6.4 Local Mesh Adjustment

The above algorithm works very well under most circumstances, but degenerate triangles that are long and thin may still occur. These triangles are undesirable since they do not represent local surface shape well and are often the cause of self-intersection of the mesh surface. Currently, we use a simple algorithm that checks for pairs of such triangles and rearrange the triangle configuration locally. After each subdivision, we check for triangles that are thin and long, and if two such triangles share an

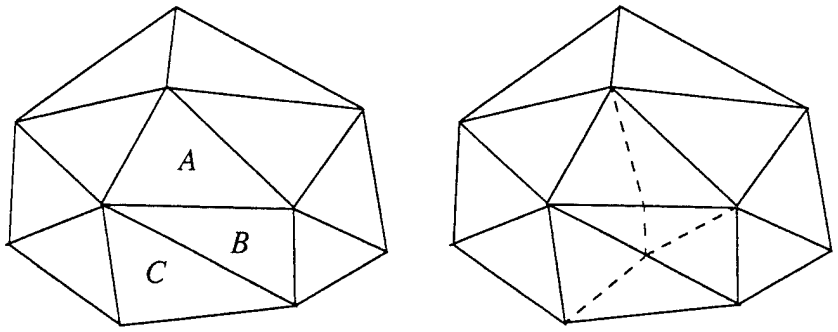


Figure 2.4 Subdivision of triangle mesh. (a) before A is subdivided, (b) after A is subdivided and the subdivision is propagated to both B and C.

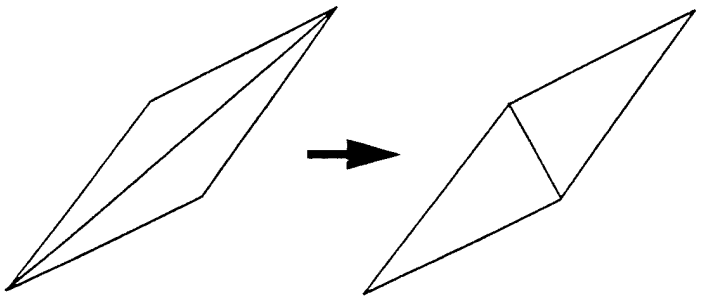


Figure 2.5 Rearranging the local configuration to eliminate long and thin triangles

edge that is the longest for both triangles, then we simply switch the cross edge as shown in Figure 2.5 .

2.7 Description of the Algorithm

In this section we give a brief description of the entire algorithm of our approach. A discussion on how to set the system parameters will follow. We assume that registered range image views of the object to be modeled are available, although we believe the algorithm can be adapted to other types of 3-D input.

We start with selecting an initial point inside the object and constructing an icosahedron shell [13] at this location. The selection process is currently done by hand and the size of the shell should be small enough so that it is completely inside the object. Since the algorithm does not depend on the actual location of the initial shell, as

long as it is inside the object, an alternative to manually selecting the initial position is to choose a smooth patch in any range image and place the shell under the patch. The system algorithm can be described as follows.

2.7.1 Algorithm 3

Let all the triangles on the initial mesh be front F_0 and push it into the front queue Q, then until queue is empty, do the followings repeatedly:

1) $F \Leftarrow$ top of the queue Q, pop the queue.
2) Subdivide the triangles in F if appropriate (see next section).
3) For each vertex $V_i \in F$, whose 3-D coordinates at time t is v_i^t, do
   a) compute the internal force $\boldsymbol{g}_i$ and external force $\boldsymbol{f}_i = \boldsymbol{h}_i$ based on Equations (2.4) and (2.5).
   b) compute the new vertex location $v_i^{t+\Delta t}$ for the current iteration according to Equation (2.3).
   c) compute prospective correspondence point of v_i, which is the intersection w_i of the surface and the line through v_i and in the direction of the mesh normal at v_i (see below).
   d) if $\left\|\boldsymbol{v}_i^{t+\Delta t} - \boldsymbol{v}_i^t\right\| > \left\|\boldsymbol{w}_i - \boldsymbol{v}_i^t\right|$, then $v_i^{t+\Delta t} \Leftarrow w_i$ and mark vertex V_i anchored.
4) For each $V_i \in F$, update its position with the corresponding new positions $v_i^{t+\Delta t}$.
5) Discard triangles from F that have thus become anchored (section [2.6]).
6) if $F = \{\emptyset\}$ then go to 1.
7) recompute connected triangle regions in F and push them into Q. Go to 1.

In step (3)(c) above, an algorithm that computes the intersection between a 3-D line and the object surface in range image form is called for. This algorithm gives the closest intersection of a line, which passes through a given vertex point and is in the direction of the estimated local mesh surface normal at the vertex, and the object surface (point P in Figure [2.3]). This is for the purpose of estimating the distance of the vertex to the prospective corresponding points on the surface of the object (Section [2.3.1]). Details of the algorithm can be found in the appendix.

2.8 Setting up the Parameters

In our current implementation, triangles that have areas larger than a threshold S_t are subdivided at each iteration. S_t is directly related to the precision of the fit of the final mesh to the input surface data. Assuming that our goal is to approximate the object surface to have a triangle fitting error δ for surfaces with maximum curvature of $1/R_t$, S_t can be easily computed by tessellating a unit sphere of radius R_t with equilateral (or near equilateral) triangles of sizes smaller or equal to S_t. This also gives us a sample configuration of an ideal front structure when the maximum

mesh tension is achieved. Let $f_{spring-max}$ be the maximum spring force exerted onto a vertex under such conditions. The inflation force needed to overcome the spring force (in order for the vertices to move) is therefore

$$f_{inflate} > f_{spring-max} \tag{2.7}$$

The inflation force is also constrained by Equation (2.3), since once a time step and a maximum displacement per iteration are set, the allowed inflation force should then be (considering 0 spring force):

$$f_{inflate} \leq \frac{d_{max}}{\Delta t} \tag{2.8}$$

where $d_{max} = \max(\|\boldsymbol{x}^{t+\Delta t} - \boldsymbol{x}^{t}\|)$ is the maximum displacement. Since a large inflation force tends to dominate the mesh's evolution, which is undesirable, we prefer a smaller one. We choose to use the minimal inflation force as shown in Equation (2.7). We can then compute the needed inflation force amplitude k according to Equation (2.5).

Now the whole issue comes down to determining $f_{spring-max}$, d_{max} and the time step Δt. The maximum spring force is determined by the spring natural length l_{ij} and the spring stiffness which are related (Equation (2.4)). In our experiments, we have used $l_{ij} = 0$ and $c_{ij} = 4.0$. d_{max} and Δt are selected to allow the mesh to evolve smoothly and quickly relative to the object size and complexity. For all the tests in this paper, we have used $2mm$ and 0.05 respectively.

Finally, the user needs to select δ and R_t. For the purpose of simplicity, in our experiments, however, we manually set R_t and allow a fixed number of N triangles to fit the sphere with a radius R_t, which gives a nominal approximation error of about $0.6mm$ with $N = 80$ and $R_t = 10mm$.

2.9 Adaptive Local Fitting, Holes and Noise

It is also worth mentioning that our algorithm is parallelizable since the computations on each front in the queue Q are independent of each other. Furthermore, the computation for each vertex within each front is also independent during each iteration.

Another advantage that this computation structure brings us is that we can adaptively adjust system parameters independently for each front based on the information that we gather from the prospective correspondence points of the vertices in the front. For example, if we have detected that the movement of the front is virtually stopped and yet the prospective correspondence points are still certain distance away, this tells us that the preset parameter R_t in previous section is too large and we should adjust it accordingly.

Another example of such adaptation is in handling holes in data. In this case, there exist areas of the object surface that are not covered by any of the input range

images, we will not be able to find prospective correspondence points for some of the vertices in the related front. Eventually, when the rest of the vertices in the front have settled down to their correspondence points, we are left with a front for which none of the vertices have a prospective correspondence point. In such situations, the system automatically sets the inflation force to zero ($k = 0$ in Equation [2.5]), which makes the mesh reach an equilibrium state that interpolates the surface over the hole.

Another important issue is the issue of noise. There are two type of noises that may affect out results. One is the noise introduced by the small misalignment among the range images. The other is the spontaneous outliers from each range image. Our system is very stable with respect to both types of noise. The first one is effectively solved by the weighted sum line-surface intersection algorithm (see Appendix) since the misalignment causes the actual intersections to form a cluster. The second type of noise usually cause the intersection algorithm on the related range image to fail to converge, in which case it does not contribute to the result of the intersection. Even if the noise does produce a wrong intersection, it can easily be filtered out as an outlier that does not belong to the correct cluster.

2.10 Test Results

We now present examples of our system in modeling a telephone handset (Phone) and an automobile part (Renault) using 20 and 24 range image views respectively, along with two examples for simpler objects. The range images are acquired using a Liquid Crystal Range Finder (LCRF) [9], and then registered using the range image registration algorithm described in [1]. Some sample range images used in the experiments are shown in Figure 2.6 . Figure [2.7] shows two examples of the balloon model in fitting two simple objects: the Wood Blob and the Tooth. In Figure 2.8 , the final rendered views of the constructed model of the Phone are shown below the wireframe drawing. Figure 2.9 shows a wireframe and the rendered image of the Renault part. The final model for the Phone has 1694 vertices and 3384 triangles, the Renault part has 2850 vertices and 5696 triangles. The total run time excluding registration on a Sun Sparc-10 running Lucid Common Lisp version 4.0 is 16'17" for the Phone and 32'26" for the Renault. Both the Phone and the Renault part measure about $200mm$ across their longer sides. Note that the wireframe drawings in the presented results are not produced using a hidden-line elimination algorithm, which is the cause of most of the spurious triangles seen in the wireframe drawings, including the "defects" in the middle which actually corresponds to a step at the back of the object.

As can be seen from the results presented above, our algorithm works very well for both simple, compact objects, as well as non-star shaped objects with complex structures. The resulting triangulated model surfaces preserve most of the important geometry feature of the objects with evenly distributed meshes. Our initial guess are all set in the neighborhood of the center of the objects and yet our balloon can successfully grow to cover all parts of the object with complex geometric structures such as the Renault part. Also, it is hard to visualize this, but the data that we use for the

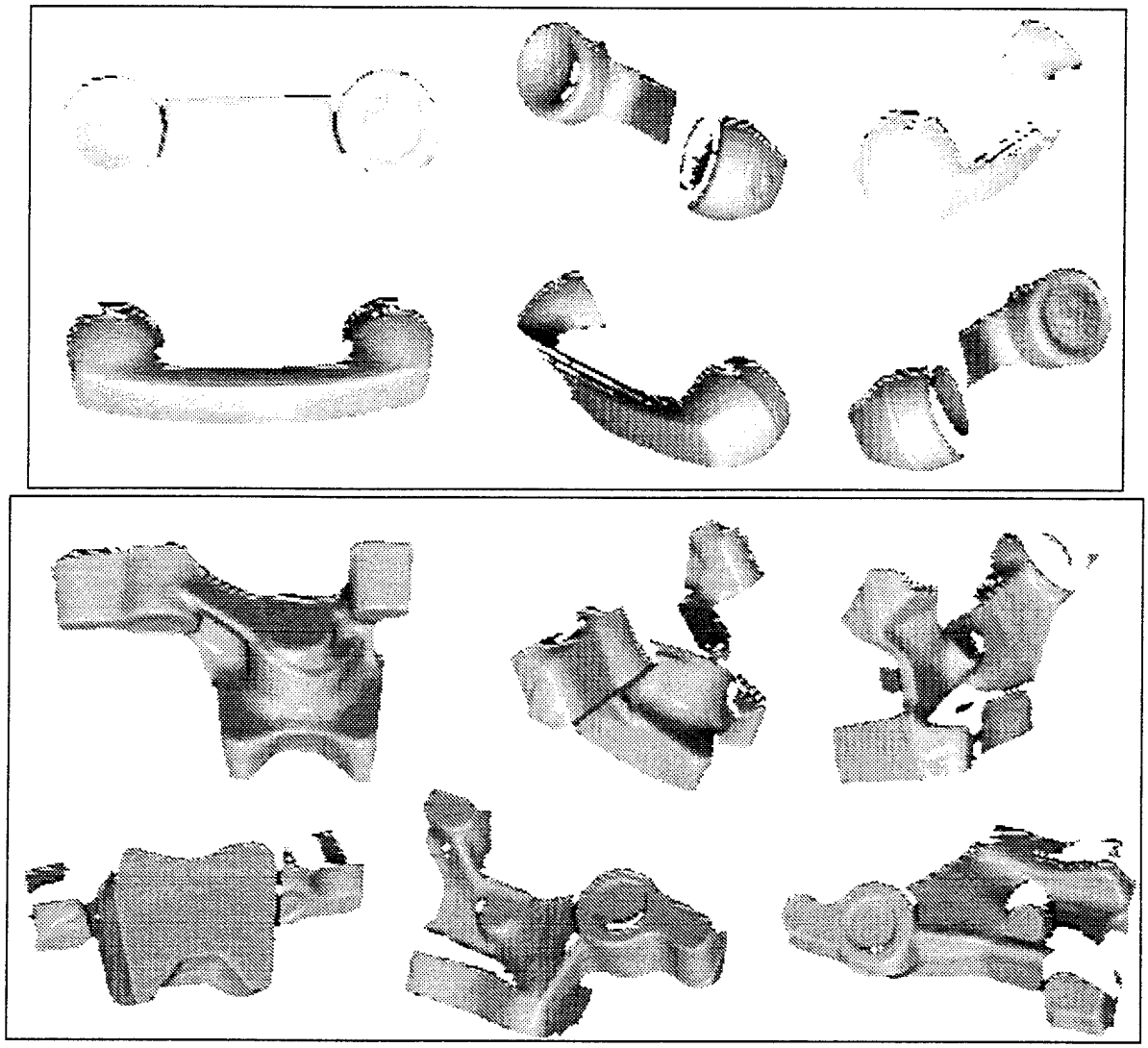


Figure 2.6 Sample range images used in constructing the Phone model and Renault model in Figures [2.8] and [2.9], shown here as shaded intensity images.

Renault part contain holes on top of both of the arms, and the resulting mesh was able to interpolate them very well. There is, however, a defect under the right arm of the Renault part, as can be seen in the wireframe drawing. It is a small opening in the mesh that tends to self-intersect which is caused by a small narrow ridge section (about $8mm$ thick, much smaller than 2 times R_t, where $\Re_t = 10mm$). We believe that this can be solved by examing and identifying local surface changes more closely and adjusting system parameters accordingly in that area.

2.11 Conclusions and Future Research

We have presented a surface description method based on a dynamic balloon model using a triangular mesh with springs attached to the vertices. The balloon

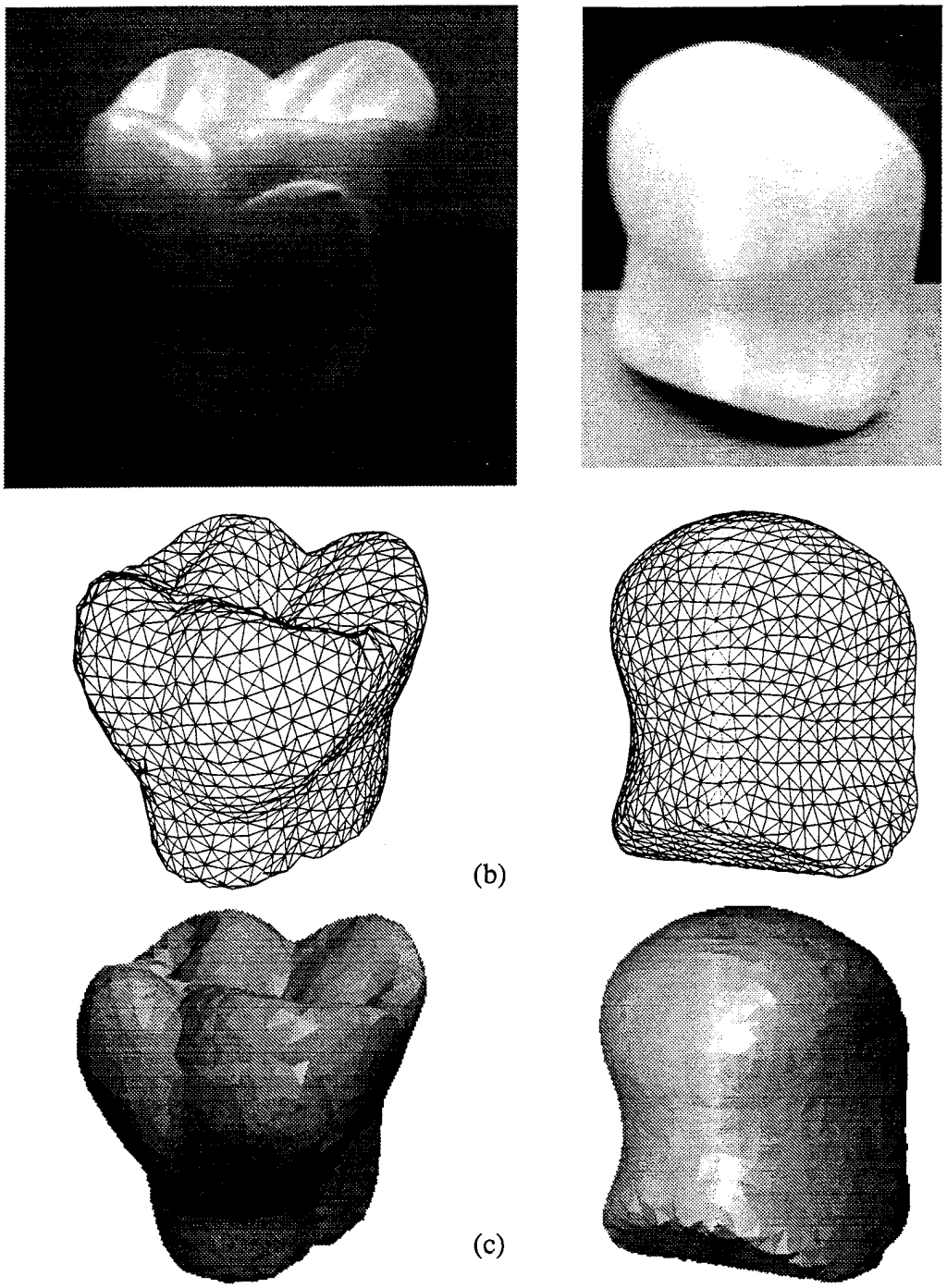
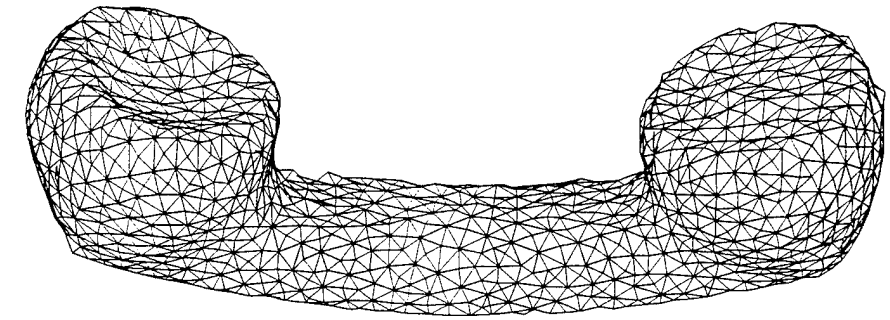
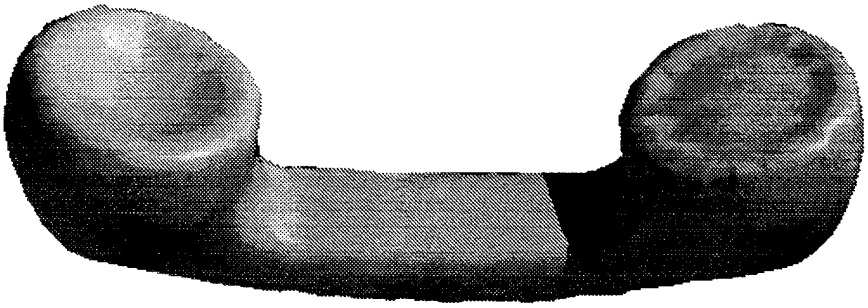


Figure 2.7 Examples of the balloon model for fitting simple objects: (a) the original intensity images of the objects, (b) the wireframes of the obtained balloon models and (c) the rendered shaded images of the models.

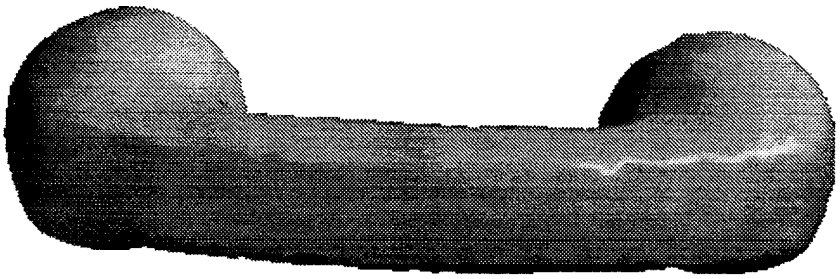
model is driven by an applied inflation force towards the object surface from inside of the object, until all the triangles are anchored onto the surface. The model is a physically based dynamic model and the implementation of the algorithm is highly paral-



(a)



(b)



(c)

Figure 2.8 The final balloon model for the Phone: (a) wireframe (b), (c) smoothly shaded.

lelizable. Furthermore, our system is not a global minimization based approach and

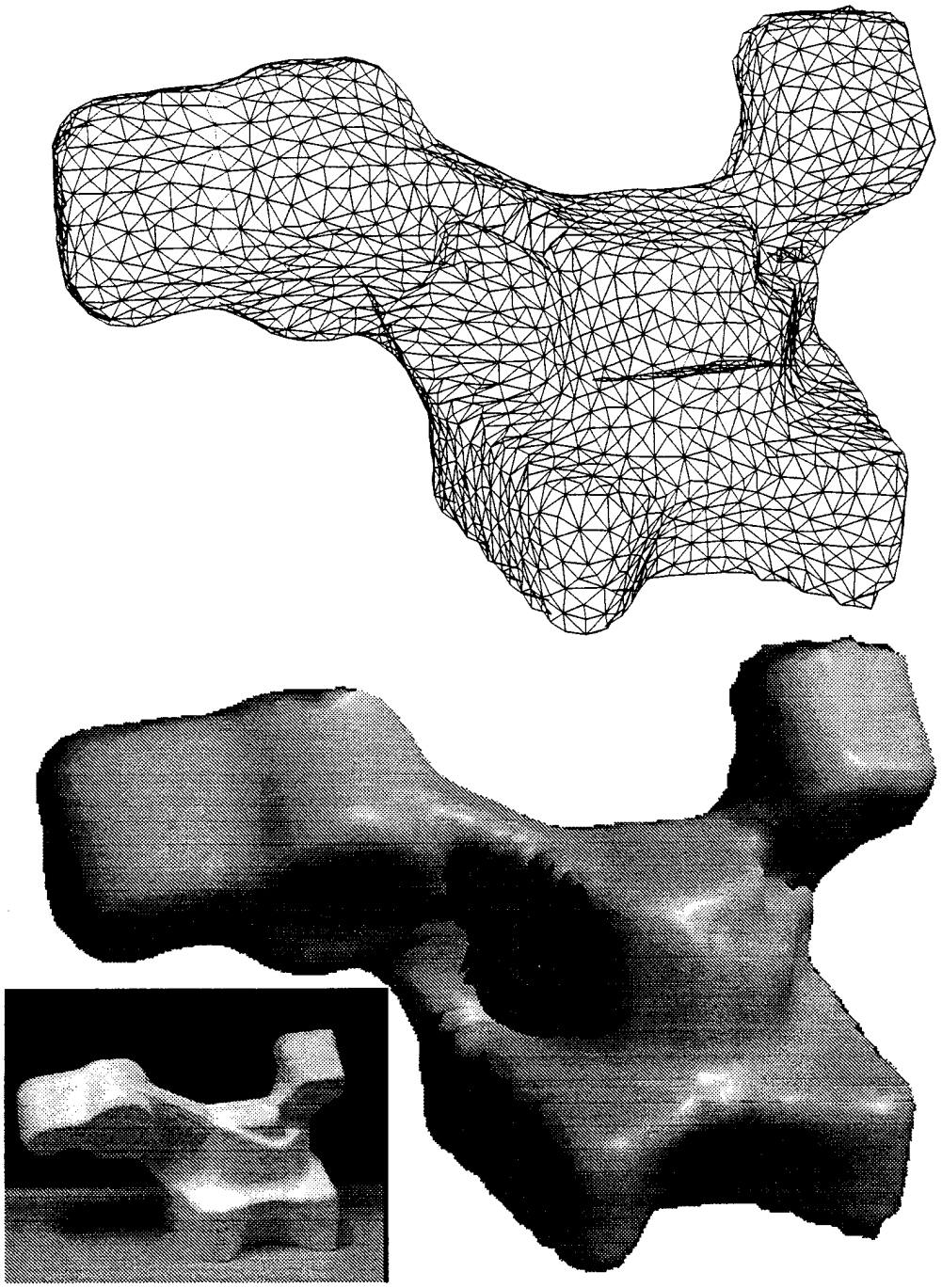


Figure 2.9 The wireframe and the rendered image of the reconstructed model for the Renault automobile part. The inserted picture is the intensity image of the actual object. Note that the wireframe is not produced by a hidden-line removal algorithm (see Section Chapter 2 on page 17).

can allow the model to adapt to local surface shapes based on local measurements. Tests showed very good results on complex, non-star-shaped objects.

As stated earlier, our goal is to achieve a correct mapping of a triangulated mesh to the surface of an object, hence there are still many ways to improve the resulting model we have achieved, including using the algorithms presented in [2] to improve triangle fitting errors, or the method in [10] to merge small triangles into larger ones without affecting the fitting error for constructing a hierarchical representation. Local smooth patches can also be constructed for high level surface property analysis. Alternative surface models, such as a smooth finite element surface model ([7]), can also be used so that the implementation of the system elements can be made more precisely. In addition, our future research consists of detecting and avoiding possible self-intersections of the mesh surface.

2.12 References

[1] Y. Chen and G. Medioni, "Object Modelling by Registration of Multiple Range Images," *International Journal of Image and Vision Computing*, 10(3):145–155, April 1992.

[2] Y. Chen and G. Medioni, "Surface Level Integration of Multiple Range Images", in *Proceedings of the Workshop on Computer Vision for Space Applications,* Antibes, France, September 1993.

[3] L. D. Cohen and I. Cohen. Finite-element Methods for Active Contour Models and balloons for 2-D and 3-D Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1131–1147, 1993.

[4] H. Delingette, M. Hebert, K. Ikeuchi, "Shape Representation and Image Segmentation Using Deformable Surfaces," *CVPR 1991* pp.467-472.

[5] Andre Gueziec, "Large Deformable Splines, Crest Lines and Matchings", *Proceedings of the International Conference on Computer Vision,* pp. 650-657, Berlin, Germany, May 1993.

[6] W.-C. Huang and D. B. Goldgof. Adaptive-Size Meshes for Rigid and Nonrigid Shape Analysis and Synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):611–616, June 1993.

[7] T. McInerney and D. Terzopoulos, "A Finite Element Model for 3D Shape Reconstruction and Nonrigid Motion Tracking", *Proceedings of the International Conference on Computer Vision,* pp. 518-523, Berlin, Germany, May 1993.

[8] M. Cecilia Rivara, "Algorithms for Refining Triangular Grids Suitable for Adaptive and Multigrid Techniques", *International Journal for Numerical Methods in Engineering*, Vol. 20, pp. 745-756, 1984.

[9] K. Sato and S. Inokuchi, "Range-Imaging System Utilizing Nematic Liquid Crystal Mask," In *Proceedings of the IEEE International Conference on Computer Vision*, pages 657–661, London, England, June 1987.

[10] M. Soucy and D. Laurendeau, "Multi-resolution surface modeling from range views," In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 348–353, Urbana-Champaign, IL, June 1992.

[11] D. Terzopoulos and D. Metaxas. Dynamic 3D Models with Local and Global Deformations: Deformable Superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, July 1991.

[12] D. Terzopoulos and M. Vasilescu, "Sampling and Reconstruction with Adaptive Meshes" *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp.70-75, Maui, HI, June 1991.

[13] M. Vasilescu and D. Terzopoulos, "Adaptive Meshes and Shells: Irregular Triangulation, Discontinuities, and Hierachical Subdivision," *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 829-832. Urbana-Champaign, IL, June 1992

2.13 AppendixThe Line-Surface Intersection Algorithm

We describe an algorithm for computing the intersection of a directed line (a ray) and the object surface represented by a set of registered range images. This algorithm is based on the line-surface intersection algorithm in [1], which computes the intersection between a line and a digital surface represented by a single range image.

We begin with a brief description of the original algorithm. As shown in Figure 2.10 , we are given a directed line l that passes through a certain point $\boldsymbol{p}$, the intersection, $\boldsymbol{q}$, of l and the surface Q can be computed as follows. We first project $\boldsymbol{p}$ onto Q in its image space and find the tangent plane of Q at the projection. Then we compute the intersection $\boldsymbol{q}^{\boldsymbol{0}}$ of the plane and l, which becomes the first approximation of the intersect we are looking for. We repeat the process by projecting the intersection approximation $\boldsymbol{q}^{\boldsymbol{k}}$ onto Q at each iteration. When this process converges, the resulting intersection is taken as $\boldsymbol{q}$. Note that in this approach, we also have a directional constraint for the found intersection, which states that the local surface normal at the computed intersection point must be within 90° of the direction of the ray, which is the direction of the mesh surface normal in this paper.

When we have more than one range images, the intersection of the line with all the range images are computed. Let $\{Q_i\}, i = 1...m$, be the set of range images and l be the line in consideration. The intersection of $\boldsymbol{l}$ and the surface $\{Q_i\}$ can be defined as the weighted sum of all the intersections:

$$\boldsymbol{q} = \frac{\sum_{i=1}^{m} a_i w_i \boldsymbol{q}_i}{\sum_{i=1}^{m} a_i w_i}, \qquad w_i = (\hat{\boldsymbol{n}}_{q_i} \cdot \boldsymbol{n}_s) \tag{2.9}$$

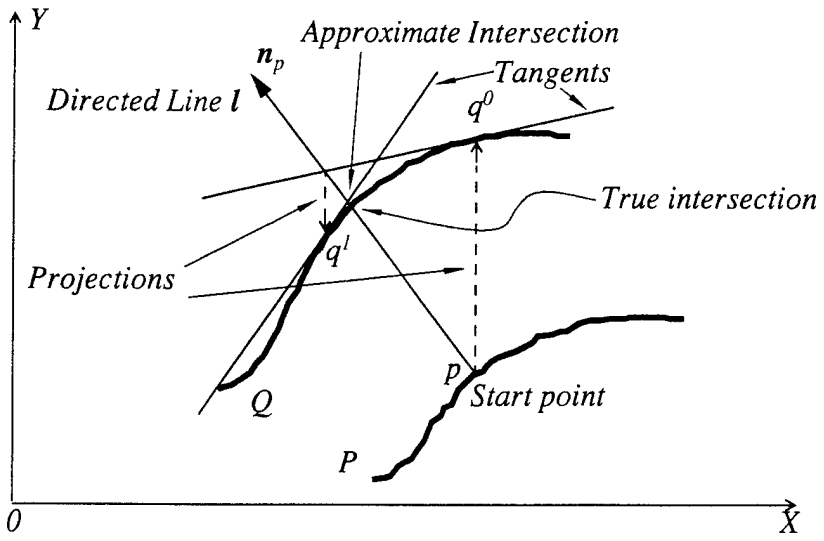


Figure 2.10 Intersecting a line with a digital surface illustrated in a 2-D case.

where $\boldsymbol{q}_i$ is the intersection of $\boldsymbol{l}$ and Q_i, $\hat{\boldsymbol{n}}_{q_i}$ is a unit vector normal to Q_i at $\boldsymbol{q}_i$, $\boldsymbol{n}_s$ is the vector pointing towards the sensor and a_i is a binary number depending on the intersection of $\boldsymbol{l}$ and Q_i, and

$$a_i = \begin{cases} 1, & \text{if the intersection exists} \\ 0, & \text{if there is no intersection} \end{cases}$$

The reason behind taking a weighted sum of the intersection points is that the sensor measurement of the position of a surface point is less reliable if the local surface is facing away from the sensor. The weight w_i is a reflection of this heuristic. In general, the sensor direction information $\boldsymbol{n}_s$ is available from the range image sensor setup and calibration. For a Cartesian range image (depth map) without sensor information, we can simply take $\boldsymbol{n}_s = (0, 0, 1)^T$.

If there are more than one intersections between $\boldsymbol{l}$ and the object surface, we can perform a clustering to separate the intersections into groups corresponding to each real intersection, and choose the closest cluster to compute the above weighted sum. In practice, we have not had to use such a clustering algorithm. This is because the result of the intersection algorithm depends on an initial point $\boldsymbol{p}$ (which is the vertex point in this paper). When the vertices are far from the object surface, $\boldsymbol{q}_i$ can be from any clusters. But we are less concerned with the actual location of the intersection at the time. As the mesh grows and the vertex get closer to the surface, almost all the intersections computed are from the closest cluster, since it is the closest local minimum when considering the intersection process as a minimization. For robustness purpose, we have implemented a simple filtering scheme to eliminate gross outliers in the intersections based on the distribution of the intersections found.

3 Surface Approximation of Complex 3-D Objects

Chia-Wei Liao and Gérard Medioni

Our goal is to generate a surface description of complex objects with parts and holes. We start by fitting a surface, assuming the object is of Genus 0, then analyze the result to further segment the description.

In the first part of our algorithm, the system provides an initial estimated surface which is subject to internal forces (describing implicit continuity properties such as smoothness) and external forces which attract it toward the data points. The problem is cast in terms of energy minimization. We solve this non-convex optimization problem by using the well known Powell algorithm which guarantees convergence and does not require gradient information. The variables are the positions of the control points. The number of control points processed by Powell at one time is controlled. The process is controlled by two parameters only, which are constant for all our experiments.

The above approach is not sufficient for complex objects with cavities, or for more than one object. We therefore propose an approach that can apply simultaneously more than one curve or surface to approximate multiple objects. Using (1) the residual data points, (2) the bad parts of the fitting surface, and (3) appropriate Boolean operations, our approach is able to handle objects more complicated than Genus 0 or with deep cavities, and can perform segmentation if there is more than one underlying object.

3.1 Introduction

Range sensing is a mature technology, and there are many methods, such as time of flight and MRI, collect 3D data based on this technology. In addition to this, 3D data can also be obtained in passive ways like stereo and shape from X methods. The data obtained from the above sources is in the form of points.

But, in computer vision, what we need are some properties such as the curvature, normal, and principal directions. These quantities relate to the underlying surface, which is not made explicit in the original data. Furthermore, it is even more difficult if some ordering relation among the data points is not known. This happens mainly when we gather data points from various sources. Analytical surface construction of a cloud of points (boundary points of the object) becomes important because it is much easier to extract the features from an analytical surfaces. So, we need some tools to construct an analytical description (for example, surface) for the collected 3D

data. A deformable model, which can give a analytical surface representation over a cloud of 3D data points, is able to serve this purpose.

The idea of fitting data by a deformable model can be found in the work of Kass *et al.* [14] in 2D. Such models are generalized in 3D by the same authors [15] for a surface of revolution.

There have been many works [16]-[29] derived from this seminal idea. But most of them either require many parameters, or cannot guarantee convergence (partially due to the use of gradient descent to minimize the energy). Furthermore, they suffer from the following problems:

First, there may be more than one underlying object, and these objects might be close to one another. Most deformable model algorithms assume that there is only one object, that is, the segmentation has been done beforehand. It takes sophisticated segmentation to separate these mixed objects, and it is often the case that segmentation is much more difficult than the fitting process.

Second, they cannot handle very well patterns with deep and narrow cavities. To capture a cavity directly through energy minimization, we need to differentiate between the data points belonging to the cavities from the other points when defining the external energy. But this might lead to a circular problem.

Third, without prior knowledge, most of them are insufficient for objects more complicated than Genus 0.

Our proposed algorithm can deal with these problems appropriately. There are mainly two parts in our algorithm. In the first part, we focus on the Genus 0 surface fitting. We apply a tested numerical method which guarantees convergence. Through an coarse-to-fine approach and a partitioning scheme, the computational time is kept in check. By using the surface fitting algorithm in the first part and appropriate boolean operations, we develop another method that can accomplish data segmentation and handle objects with deep cavities or more complicated than Genus 0.

3.2 Part 1: Genus 0 surface fitting

3.2.1 Issues

Several problems come with the algorithms dealing with deformable models:

1. Huge computational time and space: Assuming M×N control points on the fitting surface, there are 3MN variables for this surface. Theoretically, we can just inject these 3MN variables into a minimization algorithm to minimize the energy of the fitting surface. This approach turns out to be impractical due to the unbearable computational time when 3MN is large. Furthermore, most minimization algorithms need a matrix of size (3MN)×(3MN), which also results in huge space complexity. An adaptive approach can just alleviate these problems, but cannot avoid these problems in the worst case, especially when all control points or patches are bad.

2. Constructing a smooth closed surface from the B-spline control points: we obtain a closed surface by closing the top and bottom parts of the rectangular mesh as depicted in Figure 3.1 . The problem is that the poles are not smooth when we try to construct a smooth surface directly from the control points. It is because these two poles are shared by many degenerate patches (triangular patches) around them.

3. A good approximation of the data as the initial surface: the quality of the fitting result depends on the initial surface, and we would like the result to be invariant to the initial surface as long as the initial surface is not too bad.

4. The convergence of the surface to the data points: the convergence of the fitting process should be guaranteed. By using the Powell [30] minimization routine, the convergence is guaranteed.

All problems listed above are well handled by our algorithm for the surface fitting.

3.2.2 Algorithm

Now, we define some terms for further use. We define a Cap to be the triangular patches formed by a Pole and its adjacent control points. So, we always have two Caps. A Meridian (a line of constant u in parameter space) is defined to be the line connecting the two Poles, as depicted in Figure 3.1 .

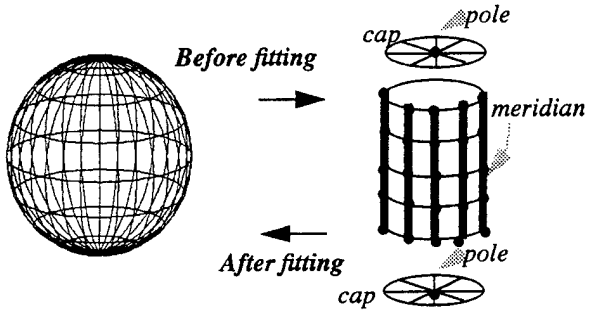


Figure 3.1 The initial closed surface and the definitions of Pole, Meridian, and Cap

A flowchart of our algorithm is in Figure 3.2 .

In our algorithm, we consider a sphere as composed of three parts, which are two caps and an open cylinder as shown in Figure 3.1 . These three parts are processed separately in our algorithm.

Instead of injecting all MxN control points into the minimization procedure (which is possible but extremely expensive), we decompose the problem into a curve fitting problem followed by a (simpler) mesh fitting problem.

Given that the caps are already in place, we select every other meridian and move their (M-4) control points, which are not shared with the two caps, to minimize their energy. We then select the remaining meridians and move their (M-4) control

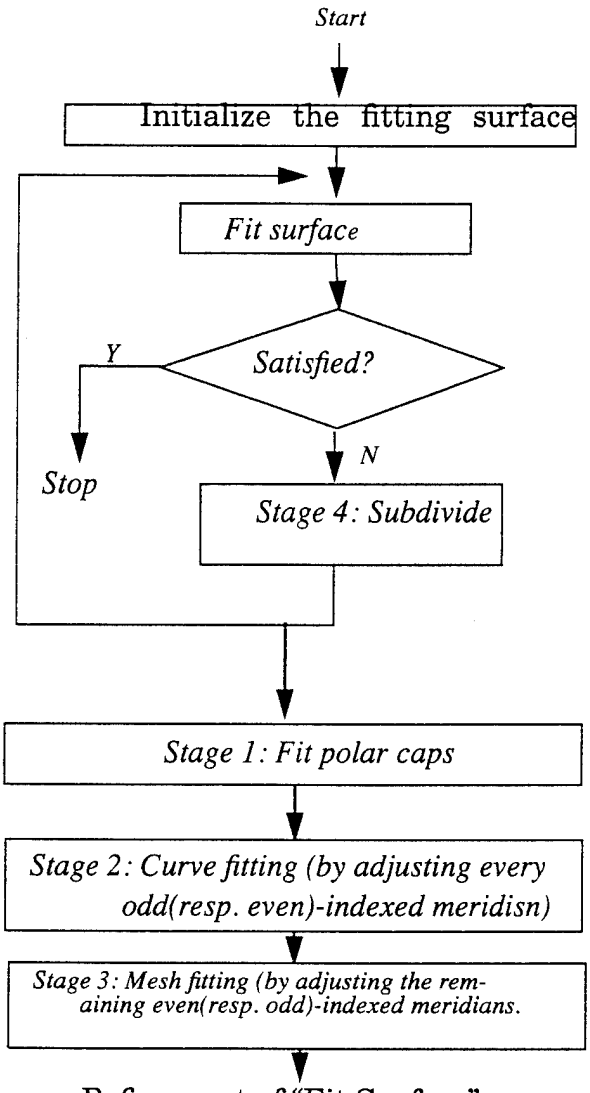


Figure 3.2 The flowchart of the Genus 0 algorithm

points, which are not shared with the two caps, to minimize the energy of the related patches this time. It is important to note, however, that only the bad segments (patches or curves) are injected into the minimization procedure (after the energy minimization, we can guarantee that the associated energy of the bad segments is reduced, but some segments with higher external energy might still be bad). Then, we subdivide all patches in four, and repeat the process until some terminating condition is met.

Our algorithm is a 4-stage one, and during the first three stages, Powell is called frequently for energy minimization.

First, we fit the caps to the data, and force the caps to be planar. This way, all tangent vectors in all directions at the pole are coplanar. When fitting, the cap can change its shape, subject to the planar constraint, in order to get the best fit. By duplicating the control points at the poles and this planar constraint, we can get a smooth caps around the poles when we want to construct a smoother (e.g. C^1) surface.

Second, we perform the curve fitting to some meridians to locate the profile of the target, and this is done by applying energy minimization to these meridians. We select the odd(resp. even)-indexed meridians and fit them to the data by treating them as approximating linear B-snake [16]. The only difference between a B-snake and a selected meridian lies in the internal energy. When calculating the internal energy of these meridians, we not only consider their own smoothness but also the smoothness between them and their immediate neighboring even(resp. odd)-indexed meridians (an example is depicted in Figure 3.8). Then we let them adapt to find the profiles of the target. These selected meridians are not influenced much by the fitting surface (by the internal energy) when moving, so they can detect the object more accurately. Please notice that the external energy of these selected meridians is defined on the curve without considering the surface nearby. The surface is separated by the selected meridians into independent "strips" in a way, and each strip contains an even(resp. odd)-indexed meridian. Each strip is bounded by two odd(resp. even)-indexed meridians.

Next, we fit each strip to the target. We select meridians of the other polarity, even(resp. odd)-indexed, to do the mesh fitting for each strip. We treat them as regular snakes, except that they are tuned to minimize the external energy (error) of the strip. This means the external energy is not only from the curve but also from the area it defines.

The fourth stage is subdivision (in Figure 3.3). If the fitting surface up to now is

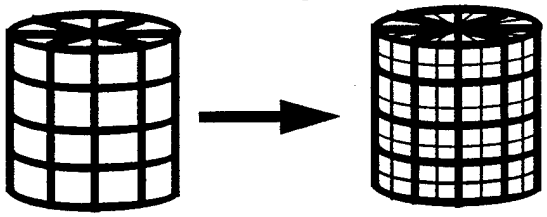


Figure 3.3 Subdivision (except on the caps).

not satisfactory, we subdivide all rectangular patches into four and then go back to stage 1, otherwise, we exit.

3.2.3 Initial guess

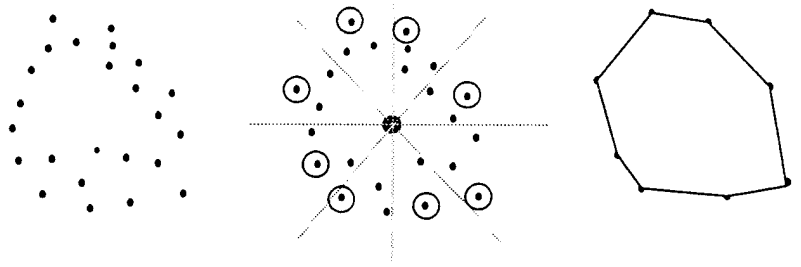
What we want to obtain is the outer contour of the target at each stage. The concavities of the objects can be detected later by applying Boolean operations applied at the next stage. We just need to have the initial surface (or curve) covering all data

points. So, the initial fitting surface would be slightly larger than the target, and it shrinks when the energy is being minimized.

First, we compute the center of mass of the data points, and extract the farthest data point in each sampled direction. The polygon formed by these extremal data points is used as the initial guess. An illustrative example is in shown Figure 3.4 .

In the 3D case, we have two alternatives. The first one is to use a cylinder covering all data points as the initial surface. In the second approach, we first calculate the center of mass C, too. In order to make the system invariant under translation and scaling, we compute the three eigen vectors of the covariance matrix of the data points, and then use these orthogonal vectors to define another coordinate system with the origin at C. We define the sampled directions according to this coordinate system, and find the farthest data point in each sampled direction. With these farthest points, we can obtain a fairly good initial estimate on the data points. In both approaches, the caps of the initial surface must be initialized to be planar. Examples for these two approaches are in Figure 3.5 and Figure 3.6 , respectively.

In both 2D and 3D cases, if there is no data point in the sampled direction, we set the corresponding radius to a predefined constant for the initial estimate of the curve and surface.



(a) Data points *(b) Center of mass and the sampled farthest data points* *(c) Initial snake (a polygon)*

Figure 3.4 Initial guess

3.2.4 Parameters

There are only two important parameters, $\text{ERROR}_{\text{threshold}}$ and $\text{RATIO}_{\text{ext-to-int}}$, set by the user in this algorithm.

$\text{ERROR}_{\text{threshold}}$ is used to determine whether or not a patch of the surface or a span of the snake is good. We only process the bad parts of the snakes and the bad patches on the fitting surface during each iteration. At each iteration, a patch (or a span) is good if its average external energy is smaller than $\text{ERROR}_{\text{threshold}}$; otherwise, it is bad.

$\text{RATIO}_{\text{ext-to-int}}$ specifies the relative importance of the external energy with respect to the internal energy. After setting $\text{RATIO}_{\text{ext-to-int}}$, two internal parameters

W_{ext} and W_{int}, concerning the weights of the external and internal energies, are set by the system. W_{ext} is always 1, and W_{int} is set as below:

$$W_{int} = \frac{E_{current-ext}}{E_{current-int} \times RATIO_{ext-to-int}}$$

Where $E_{current\text{-}ext}$ is the current external energy of the fitting surface or snake, and $E_{current\text{-}int}$ is the current internal energy of the fitting surface or snake.

Every time Powell is invoked, W_{int} is re-calculated based on $RATIO_{ext\text{-}to\text{-}int}$ and the current internal and external energies of the fitting surface. So every time, Powell may be called with different W_{int}.

The reasons why we set $RATIO_{ext\text{-}to\text{-}int}$ is that W_{ext} and W_{int} are different measures and thus on different scales. $RATIO_{ext\text{-}to\text{-}int}$ serves to normalize two energies. $RATIO_{ext\text{-}to\text{-}int}$ is always greater than 1; otherwise, the fitting surface is unlikely to conform to the data points, as we now explain.

As we subdivide the surface after each iteration, the fitting surface is approaching the real object. We have more confidence in the fitting surface. So it is suggested that the internal energy weight be reduced as the process goes on. One more advantage of $RATIO_{ext\text{-}to\text{-}int}$ is that the weight of the internal energy tends to decrease as the process goes on, because E_{ext} decreases faster than E_{int} does when $RATIO_{ext\text{-}to\text{-}int}$ is greater than 1. By setting $RATIO_{ext\text{-}to\text{-}int}$ to be a constant greater than 1, we can diminish the importance of the internal energy after each iteration implicitly, and thus obtain a better fitting surface.

On the contrary, if we set W_{int} directly and keep it unchanged, then the internal energy tends to dominate at the later iterations because the external energy decreases faster than the internal energy does. This might not lead to a good fit.

These two parameters could be constants in most cases, which means we can use the same values for most data regardless of the complexity of the underlying object (because the weight of the internal energy decreases automatically as the fitting process goes on). Thus, we can assume the underlying object is smooth, and assign a liberal weight to the internal energy. We use the same parameter values in our experiments.

3.2.5 Summary

In summary, in the first step, we fit the caps. Next, the odd(resp. even)-indexed meridians are used to find the profile or frame of the target, and the surface is divided into strips by these meridians. Then the even(resp. odd)-indexed meridians are applied to fit the strips to the data. Finally in stage four, we subdivide the surface, that is, we divide each rectangular patch at its center into four (we can avoid degenerate patches this way). We break the 3D surface problem into a set of 2D (linear) B-snake

problems in a way. Also notice that, at each step, although we have different ways of calculating the internal energy E_{int}, and the external energy E_{ext}, the basic idea is the same.

We can see that this is a typical coarse-to-fine approach. We start with few control points and large patches, then we increase the number of the patches and control points at later iterations.

Our algorithm overcomes the time and space complexities, and closed surface problems by (1) breaking the 3D surface problems into several 2D snake problems, which is shown in stages 2 and 3, (2) coarse-to-fine approach, and (3) forcing the cap to be planar. Due to the robustness of Powell, we do not need a good initial guess. Also, the Powell method guarantees convergence.

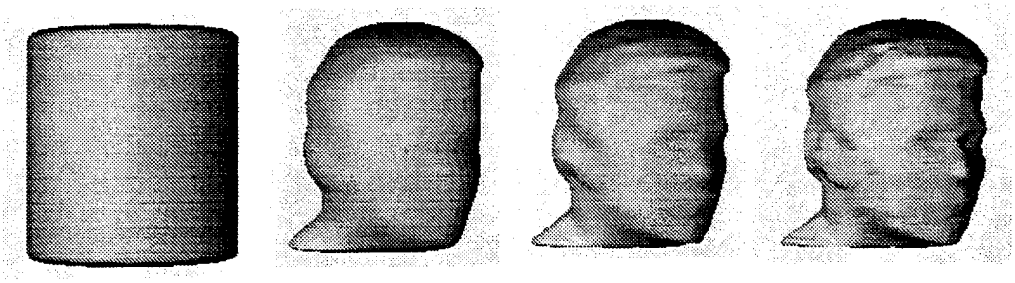
Please notice that stages 2 and 3 can be performed very fast when there are few control points. We can take advantage of this to get more reliable global information. The computational time could also be largely reduced by parallel processing. It is obvious that (1) the two caps are independent of each other, (2) all odd(resp. even)-indexed meridians are independent of one another, and (3) all even(resp. odd)-indexed meridians are independent of one another, so each stage can be performed in parallel.

3.2.6 Experiments

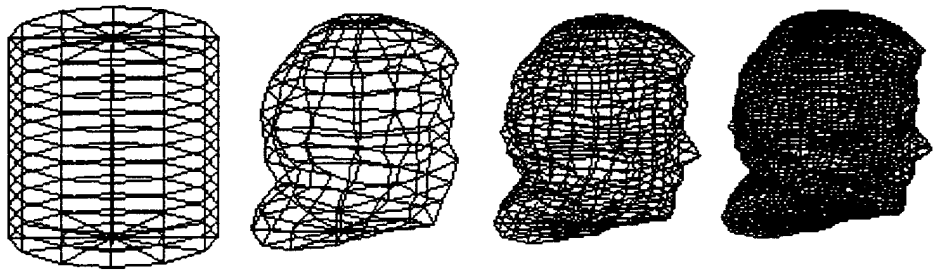
$\text{ERROR}_{\text{threshold}}$ and $\text{RATIO}_{\text{ext-to-int}}$ are 1.0 and 10 in all of our experiments. The first experiment on the head is performed on Sparc 10, and the second on IRIS Indigo.

Figure 3.5 shows the evolution of the fitting surface with the cylindrical initial surface, and both the shaded and wire-frame results are shown. The average external energy of the surface point is initially 20.15 voxels, 1.43 voxels after the first iteration, 1.21 voxels after the second iteration (one sub-division has been done), and 1.18 voxels after the third iteration (two sub-divisions have been done).

An experiment on the Renault part (see Figure 3.6). Around 3820 control points are used. The maximum point error of the fitting surface is 2.0, and the average is 0.37 voxel. The running time here includes the time for constructing the energy field, so it is longer than those for the other experiments. There are three iterations (two subdivision). A 200×200×200 cube is used to store the external energy. The average error is 10.54 voxels initially, 1.04 voxels after the first iteration, 0.41 voxel after the second iteration, and 0.37 voxel finally. The distribution of the error is also shown. (a) shows the original data. (b) is the shaded result. (c) is the initial surface. (d), (e), and (f) are the results after each iterations. (g), (h), (i), and (j) show the patches with an error above 0.3, 0.6, 0.9, and 1.2, respectively. There are only three patches with an error over 1.2 (and less 2.0). The detailed information is on Tables 1 and 2.



(a) Initial surface. *(b) Result 1* *(c) Result 2* *(d) Result 3*



(e) Initial wire *(f) Wire frame 1* *(g) Wire frame 2* *(h) Wire frame 3*

Figure 3.5 The evolution of the experiment of head 1. (a) is the initial surface (cylinder). (b), (c), and (d) are the deformed results for each iteration. The surface has been sub-divided twice. (e), (f), (g), and (h) show the wire frames of (a), (b), (c), and (d).

3.3 Part 2: Surface fitting for complex objects

3.3.1 Issues

Most deformable algorithms are deficient when (1) there are multiple underlying objects, (2) there are deep cavities, or (3) the underlying objects are more complicated than Genus 0. These are the problems we want to resolve here.

Table 1: Performance on the Renault part experiment

	Initial surface	first iteration	second iteration	third iteration
Error	10.54	1.04	0.41	0.37
Control points	16×18	16×18	32×33	64×63
Computation time (min.)	0.1	21	19	3

Table 2: General information on the Renault part experiment

Number of data points	Time for constructing energy field	No of subdivisions
214100	2.16 minutes	2

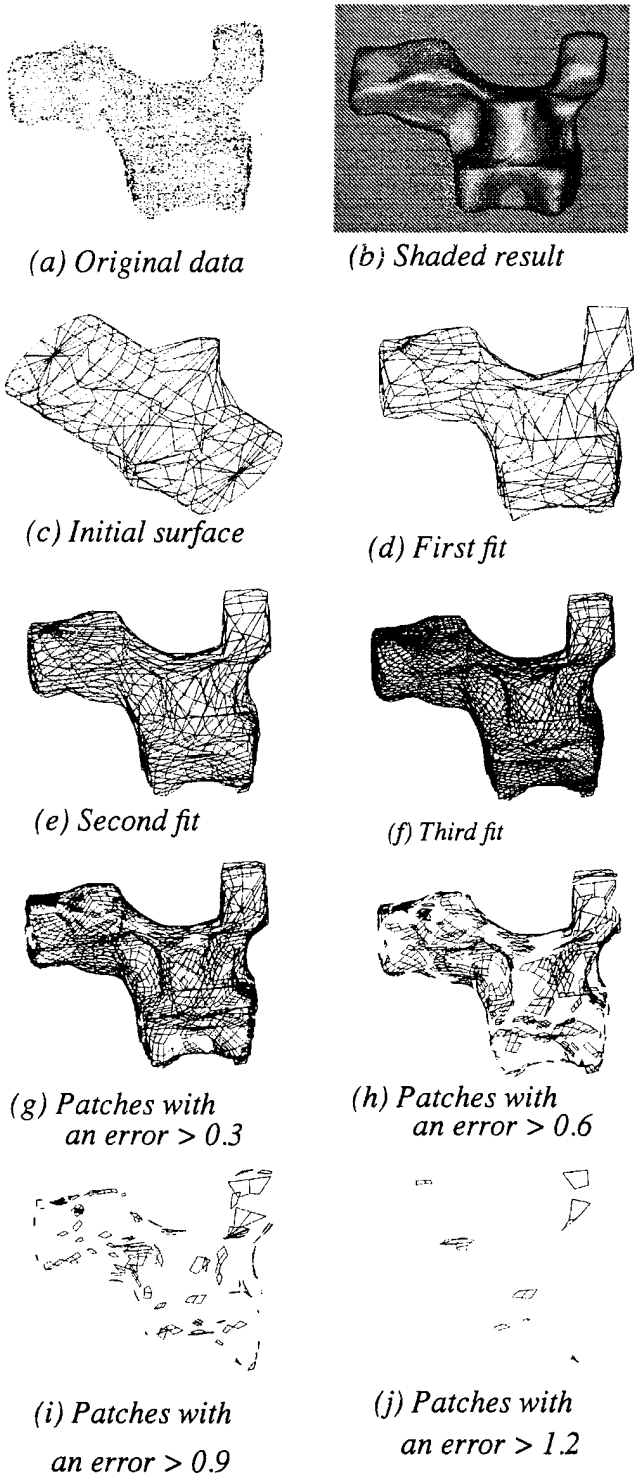


Figure 3.6 Experiment on a Renault part.

3.3.2 Algorithm

Our proposed approach is to use a *hierarchy*. We would like to handle a complicated object by representing it as a tree. The underlying object can be obtained by applying Boolean operations recursively to the tree. Every node in the tree, including the root, is supposed to be a simple object without narrow cavities or inner tunnels, that is, each node contains the outer contour of some object. We still use the energy fields mentioned in the previous section to detect the outer contour.

Let B be the outer contour first found. Then, we isolate residual data points that are not well fitted, and cluster these residual data points into groups. Next, we find those bad parts of the fitting curve with high external energy. For each bad part, we check if there is a group of residual data points connected with it. If so, we merge it into this group, otherwise we consider this bad part good because no data points are nearby. Now, we have groups of points. We treat each group of bad data points as an object and find out its contour recursively. Let P be one of the contours.

If P is inside B, which means P is a negative part of B, then $B = B \setminus P$;

If P is outside B, which means P is a missing part of B, then $B = B \cup P$.

How do we check if sub-part P is inside or outside body B? Because the boundary of any object here is a closed continuous B-spline curve, we can separate the inside from the outside by setting different gray levels inside these two regions. Through the gray level values, we can tell whether a pixel is in B or not. Thus we can determine if P is inside or outside B easily.

An illustrative 2D example is given in Figure 3.7 . (a) shows a complex object

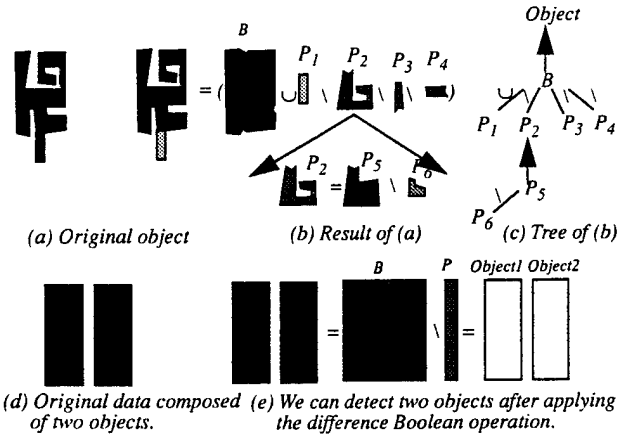


Figure 3.7 Two illustrative examples of object decomposition.

with deep cavities. In (b), by finding the outer profile only, and using the residual data

points and the bad curve segments, we get simple-shaped primitives B, P_1, P_2, P_3, P_4, P_5, and P_6. The original object can be restored by applying the appropriate union and difference Boolean operations. (c) shows the tree associated with the result in (b). (d) shows two objects close to each other. By applying the difference operation, we can classify this set of data points into two different objects as shown in (e).

3.3.3 Experiments

In the first experiment (in Figure 3.8), we use hand-made data, which consists

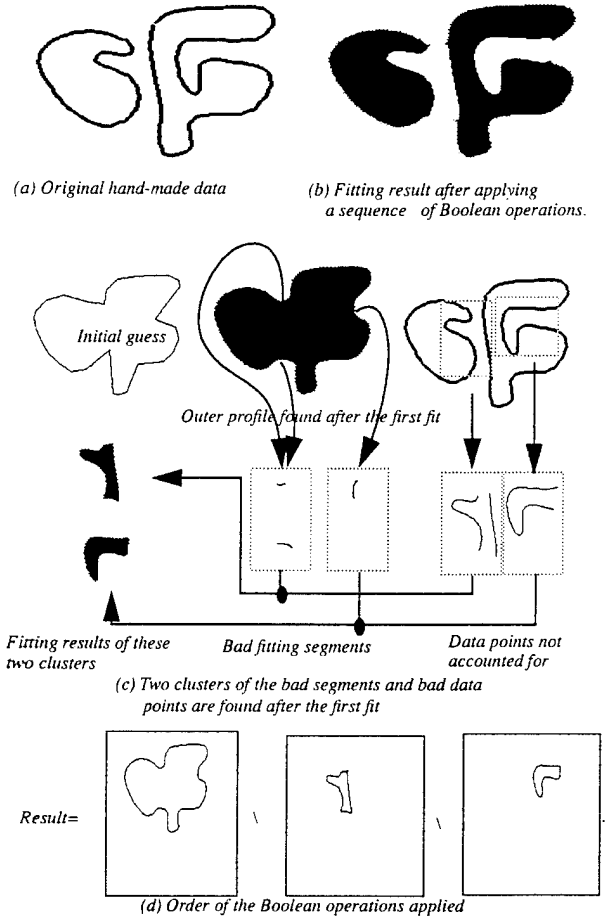


Figure 3.8 Segmentation on two concave patterns.

of two simple objects with deep concavities. After applying a B-snake and appropriate Boolean operations, these two objects are differentiated. (a) is the original data, and (b) is the final result. (c) shows the initial guess, and how the residual data points and the bad B-snake segments merge into two clusters (which form the negative parts of the target). (d) shows the Boolean operations applied. At first, the outer contour is found. We find two clusters of residual data points inside the outer contour, so the dif-

ference Boolean operations are applied. Finally, the original contour is separated into two

In the second experiment (in Figure 3.9), the data points are the same as those

(a) Data corrupted by noise *(b) Fitting result*

Figure 3.9 Experiments on noisy data.

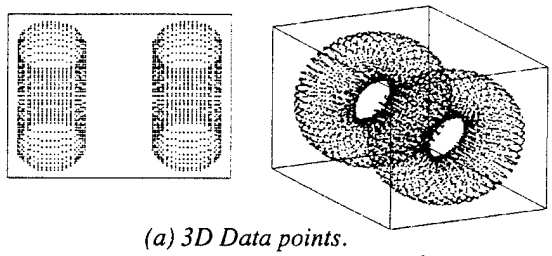
in Figure 3.8 except that (1) half of the data points are deleted randomly, and (2) the data points are randomly shifted at most 3 pixels. (a) shows the data points, which have broken boundaries. (b) is the result. The sequences of the Boolean operations applied are exactly the same as those in the previous experiments. The boundaries detected here are more irregular, but they are still continuous B-spline curves. So the objects and the hole can still be correctly segmented.

The third experiment, in Figure 4.10 , is on 3D data which is composed of two separate genus 1 toruses. (a) shows the data points, (b) is the result (object A) after the first fit, which results in a dumbbell-like shape, (c) is the residual of the data points that are not accounted for by object A. They are from the inner parts of the two toruses, and (d) shows the bad parts of object A without data points nearby. They are from the two ends and middle of object A. (e) is the merger of points in (c) and (d). (f) is the fitting result (object B) to points in (e). Because object B is inside object A, a difference operation A\B is applied, which leads to two separate entities. (g) shows the shells of the two separate entities, which are toruses. In this experiment, objects (two separate toruses) more complex than genus 0 are well handled, and the data segmentation, which segments the data points into two parts, is automatically done after fitting.

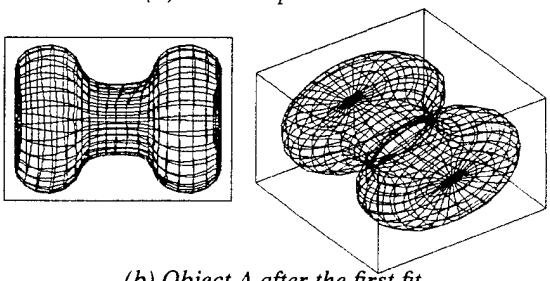
3.4 Discussion

There are several important aspects in this paper:

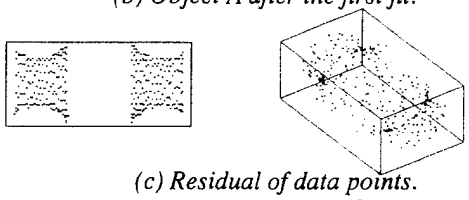
- Our new scheme is a coarse-to-fine approach. It divides all patches after each iteration. It is efficient because if a patch is really good, then the only operation applied to it in the future is just sub-division, which costs very little. This scheme also preserve the rectangular structure of the surface after each sub-division, which makes generating smooth surface easier and cheaper. This approach is free from the degenerate patch problem because a rectangular patch is always divided into 4 rectangular ones. We prefer the rectangular mesh to the triangular mesh because it is much easier to construct a smoother surface from



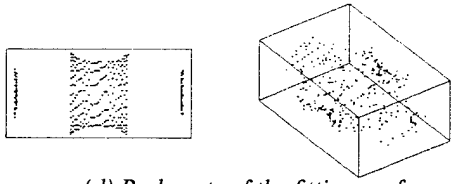
(a) 3D Data points.



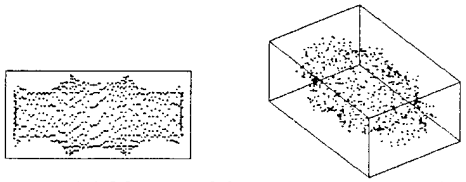
(b) Object A after the first fit.



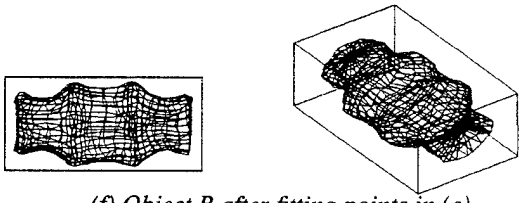
(c) Residual of data points.



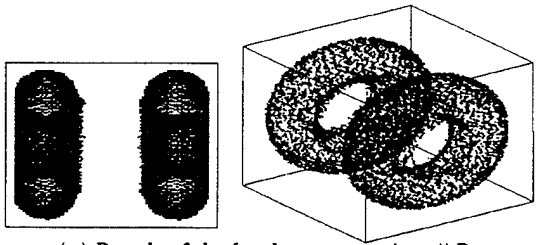
(d) Bad parts of the fitting surface.



(e) Merger of data points in (c) and (d).



(f) Object B after fitting points in (e).



(g) Result of the boolean operation A\B.

Figure 4.10 Experiment on two tori

the rectangular mesh, and the properties, such as derivatives, are much easier to obtain.

- There is always a large matrix associated with the minimization algorithm, and the size of the matrix is proportional to the square of the number of the variables. This might result in the memory explosion if there are many control points to handle at a time. Also, the numerical method goes extremely slow under this situation. With the partitioning scheme, we break a 3 dimensional problem down into several 2 dimensional problems, and then the space and time complexities can be reduced significantly. We separate the surface into several strips, so Powell is always called with a limited number of variables. For example, if the fitting surface has MxN control points, the maximum number of variables sent to Powell is around 3*(N-4). Only the bad parts of the strips and the meridians are tuned by Powell. So, in practice, the number of variables is far below 3*(N-4). The caps only have (N+5) variables, which is also low.
- We reduce the weight of the internal energy implicitly as the iteration goes on, because we have more confidence in the fitting surface after each iteration. This way, the discontinuities of the data can be well preserved.
- We use the Powell minimization routine which is more stable, robust, and accurate than the gradient descent approach.
- Due to the independency among the caps and meridians, our algorithm could run in parallel.
- This system is easy to control because there are only two global parameters to adjust.
- The assumptions of (1) one underlying object only, (2) the availability of good initial guess, and (3) geometrically simple objects without deep cavities have been the weakness points of the deformable model algorithms. By applying multiple snakes simultaneously and Boolean operations, objects can be segmented into independent ones, and cavities can also be well handled. Our algorithm makes the deformable model much more versatile.

3.5 Future work

We would like to upgrade all algorithms in this paper completely to 3D ones, and build a working system for both 2D and 3D. In addition, we would like to work out a better 3D surface representation which can handle multiple objects and objects more complicated than Genus 0.

References

[14] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models", in *International Journal of Computer Vision,* January 1988, pp.321-331.

[15] D. Terzopoulos, A. Witkin, and M. Kass, "Constraints on deformable models: Recovering 3D Shape and Nonrigid Motion", *Artificial Intelligence,* Vol. 36, 1988, pp. 91-123.

[16] Sylvie Menet, Philippe Saint-Marc, and Gérard Medioni, "B-snakes: implementation and application to stereo", in *Proceedings of Image Understanding Workshop 1990*, pp.720-726, Pittsburgh, September, 1990.

[17] Brent, Richard P. 1973, in *Algorithms for Minimization* Laurent D. Cohen, "On Active Contour Models and Balloons", in *Computer Vision, Graphics, and Image Processing*, Vol. 53, No. 2, March 1991, pp.211-218.

[18] H. Delingette, M. Hebert, K. Ikeuchi, "Shape Representation and Image Segmentation Using Deformable Surfaces", in *Computer Vision and Pattern Recognition*, 1991, pp467-472.

[19] Scott, G. L. (1987), "The alternative snake - and other animals", in Eklundh, J,-O., editor, *The 1987 Stockholm Workshop on Computational Vision*, Stockholm. Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, TRITA-NA-P8714 CVAP 47.

[20] Staib, L.H. and Duncan, J.S. (1989). "Parametrically Deformable Contour Models" in *Computer Vision and Pattern Recognition, 98-103, San Diego, CA. IEEE Computer Society Press.*

[21] Leitner, F., Marque, I., Lavallée, S., and Cinquin, O. (1990)., "Dynamic Segmentation: Finding the Edge with Differential Equations and 'Spline Snakes'". *Technique Report TIMBTIM 3-IMAG, Faculte De Medecine*, La Tronche, France.

[22] Curwen, R. M., Blake, A., and Cipolla, R. (1991). "Parallel Implementation of Lagrangian Dynamics for Real-Time Snakes", In Mowforth, P., editor, *British Machine Vision Conference,* 29-35, Glasgow. Springer-Verlag, London.

[23] Cohen, L.D. and Cohen, I. (1990), "A Finite Element Method Applied to New Active Contour Models and 3D Reconstruction from Cross Sections", in *Proc. Third International Conference on Computer Vision*, 587-591. IEEE Computer Society Conference. Osaka, Japan.

[24] Terzopoulos, D. and Waters, K. (1990), "Analysis of Facial Images Using Physical and Anatomical Models", in *Third International Conference on Computer Vision*, 727-732, Osaka, Japan.

[25] Carlbom, I., Terzopoulos, D., and Harris, K. M. (1991), "Reconstruction and Visualizing Models of Neuronal Dendrites" *in Patrikalakis, N. M., editor, Science Visualization of Physical Phenomena*, 623-638. Springer-Verlag, New York.

[26] Gabriel Taubin, "An Improved Algorithm for Algebraic Curve and Surface Fitting" in *International Conference on Computer Vision,* May, 1993.

[27] Gabriel Taubin, Fernando Cukierman, Steven Sullivan, Jean Ponce, and David J. Kriegman, "Parameterizing and Fitting Bounded Algebraic Curves and Surfaces" in *Computer Vision and pattern Recognition*, 1992

[28] Gabriel Taubin, Ruud M. Bolle, and David B. Cooper, "Representing and Comparing Shapes Using Shape Polynomials", in *Computer Vision and pattern Recognition*, 1989.

[29] Richard Szeliski, David Tonnesen, and Demetri Terzopoulos, "Modeling Surfaces of Arbitrary Topology with Dynamic Particles", in *Computer Vision and pattern Recognition*, 1993

[30] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, in *Numerical Recipes in C, The Art of Scientific Computing* (Cambridge), Chapter 10.

4 Surface Approximation of a Cloud of 3D Points

Chia-Wei Liao and Gérard Medioni

We present an implementation of deformable models to approximate a 3-D surface given by a cloud of 3D points. It is an extension of our previous work on "B-snakes" [44] and [42], which approximates curves and surfaces using B-splines. The user (or the system itself) provides an initial simple surface, such as a closed cylinder, which is subject to internal forces (describing implicit continuity properties such as smoothness) and external forces which attract it toward the data points. The problem is cast in terms of energy minimization. We solve this non-convex optimization problem by using the well known Powell algorithm which guarantees convergence and does not require gradient information. The variables are the positions of the control points. The number of control points processed by Powell at one time is controlled. This methodology leads to a reasonable complexity, robustness, and good numerical stability. We keep the time and space complexities in check through a coarse to fine approach and a partitioning scheme. We handle closed surfaces by decomposing an object into two caps and an open cylinder, smoothly connected. The process is controlled by two parameters only, which are constant for all our experiments. We show results on real range images to illustrate the applicability of our approach. The advantages of this approach are that it provides a compact representation of the approximated data, and lends itself to applications such as non-rigid motion tracking and object recognition. Currently, our algorithm gives only a C^0 continuous analytical description of the data, but because the output of our algorithm is in rectangular mesh format, a C^1 or C^2 surface can be constructed easily by existing algorithms.

4.1 Introduction

Range sensing is a mature technology, and there are many methods, such as time of flight and MRI, collect 3D data based on this technology. In addition to this, 3D data can also be obtained in passive ways like stereo and shape from X methods. The data obtained from the above sources is in the form of points.

But, in computer vision, what we need are some properties such as the curvature, normal, and principal directions. These quantities relate to the underlying surface, which is not made explicit in the original data. Furthermore, it is even more difficult if some ordering relation among the data points is not known. This happens mainly when we gather data points from various sources. Analytical surface construction of a cloud of points (boundary points of the object) becomes important because it is much easier to extract the features from an analytical surfaces. So, we need some tools to construct an analytical description (for example, surface) for the collected 3D

data. A deformable model, which can give a analytical surface representation over a cloud of 3D data points, is a good candidate for this purpose.

The idea of fitting data by a deformable model can be found in the work of Kass *et al.*[41] in 2D. Such models are generalized in 3D by the same authors [57] for a surface of revolution.

Recently, Cohen *et al.*[34] have expressed the surface fitting problem as a functional minimization problem. They enhance performance and numerical stability using variational approach and Finite Element Method. Terzopoulos and Metaxas[55] present a physically based approach to fitting complex 3-D shapes using a class of dynamic models which can deform locally and globally, and satisfy the conflicting requirements of shape reconstruction and shape recognition. Based on the elastic properties of real materials, Pentland and Sclaroff [47] propose a closed-form, physically based solution for recovering a 3-D solid model from collections of 3D surface measurements. A closed-form solution can be obtained in their system using Modal Dynamics. Nastar and Ayache's approach [46] is similar to Pentland's, but they delete a nonlinear term in the physics governing equation, and then the computation is simplified. This way they can speed up the fitting process. Terzopoulos and Vasilescu [56], motivated by concepts from numerical grid generation, use adaptive meshes that could sample and reconstruct intensity and range data. In their recent work [58], they develop some algorithms to handle the discontinuity problem, and by subdividing the adaptive meshes, reasonable results can be obtained. Delingette *et al.*[35] model an object as a closed surface that is deformed subject to attractive fields generated by input data points and features. Features affect the global shape of the surface while data points control local shape. Sinha and Schunck [51,52] use weighted bicubic splines, which are able to interpolate data with discontinuity without much distortion, as a surface descriptor. A regularized least square fit with the addition of an adaptive mechanism in the smoothness functional is applied in order to make the solution well behaved. Minimizing the energy by handling the fitting B-spline surface independently along parameters u and v and interpolating the external energy field, Gueziec [36] obtains good fitting results efficiently in terms of time and space complexities. McInerney and Terzopoulos [43] apply a dynamic balloon model and finite element method to reconstruct a 3D object. their model can give the information to measure the differential geometric properties of the fitted surface. Muraki [45] uses a "Blobby" model for the shape description. In his approach, an potential energy field is constructed through the "primitive," realized as an implicit function F(P)=T. By splitting one selected bad primitive into two at a time, and fitting them to the data, the shape of the object can be retrieved. Hoppe *et al.*[38] propose an algorithm based on the estimated tangent plane of each sampled data point, and then a Riemannian Graph is constructed using EMST (Euclidean Minimum Spanning Tree). The contour of the object can be derived from this graph. Solina's approach [53] is based on superquadrics, and several functions concerning bending, tapering, and cavity deforma-

tion, are employed to adapt the superquadrics to the data points. Han *et al.*[37] introduce hyperquadrics, which is a generalization of superquadrics, for shape recovery from range data. Their model can represent any arbitrary convex shapes. Huang and Goldgof [39] develop an algorithm similar to Vasilescu and Terzopoulos's. Their algorithm differs mainly in the way the meshes are subdivided. Instead of adjusting the stiffness of the spring to fit the local properties of the data, they update the patch size adaptively by adding or deleting nodes appropriately. A new node is added between two neighboring nodes if they are far from each other, and two neighboring nodes are merged if they are very close to each other.

To summarize, most of the algorithms described above either require many parameters, or cannot guarantee convergence (partially due to the use of gradient descent to minimize the energy). Our proposed algorithm can deal with these problems appropriately. In our approach, we apply a tested numerical method which guarantees convergence, and through an coarse-to-fine approach and a partitioning scheme, the computational time is kept in check.

4.2 Description of our approach

Most of the algorithms described earlier are suffering from long computation time and large space complexity. Furthermore, sometimes due to the instability of the numerical methods, such as gradient descent, the result might be bad because of over-shooting. It is not easy to detect over-shooting, let alone to backtrack and tune the stepsize. Most algorithms attacking this problem are based on gradient descent. Here we adopt Powell, a much more accurate and stable method. Through some mechanisms in our algorithm, the computation time is kept in check. The formalism we are about to establish amounts to deforming the initial surface to conform as closely as possible to the given 3D data points. This is achieved by defining an attraction force field around these data points to bring the initial surface closer to them. The initial surface is updated by a function minimization algorithm, Powell. Currently, the surface consists of C^0 rectangular mesh. In a word, we treat the whole process as minimization problem – given an initial guess, which may be a cylinder, we find the local minimum of the energy function by tuning the variables (the positions of the control points).

The total energy of the fitting surface is defined as below:

$$E_{total} = W_{int} * E_{int} + W_{ext} * E_{ext}$$

E_{ext} expresses the distance between the fitting surface and the data points. E_{int} depends on the constraints, such as smoothness. The definition of E_{int} is subject to change when different constraints are applied. W_{int} and W_{ext} are the corresponding weights. Without loss of generality, W_{ext} is always 1 in our system.

Once a C^0 surface is obtained, it is also possible to upgrade it to C^1 using existing algorithms [50], although we do not address this point here. Due to the convex-hull

property of B-spline function, we can also use these C^0 control points to construct a C^1 or C^2 B-spline surface without significant error when there is a large number of control points and these control points are close to one another.

4.2.1 Global parameters

In our system there are only two global parameters set at the beginning, which makes the whole system easy to control.

The first is $ERROR_{threshold}$, which is used to determine the goodness of the patch on the mesh. The second is $RATIO_{ext\text{-}to\text{-}int}$, which is used to set the initial respective contributions of the internal and external energies. These two global parameters will be explained in details in the following section. It is worth noting that all our data sets were processed with the same parameter values.

4.2.2 Surface Representation

First, how do we represent a surface? The most common primitives are triangular and rectangular meshes. The triangular mesh is more general, but suffer from the following drawbacks:

a) Eventually, we would like to construct a smooth surface, but it takes high degree polynomials to construct a C^1 or C^2 surface from the triangular mesh, which is expensive. In most cases, the algorithms for this purpose require the gradient information of each point, which does not come with the triangulation, and needs to be estimated. The rectangular mesh can be upgraded it to C^1 or C^2 easily through B-spline or Bezier functions.

b) For energy minimization, we need a method to estimate the smoothness of the surface. It is not easy to estimate the smoothness of the triangular mesh, compared to the rectangular mesh, whose derivative information can be evaluated with simple mathematics. In a way, each patch on the triangular mesh is parameterized by different parameters, which makes the estimation of the smoothness difficult.

In our algorithm, we need to sub-divide some patches on the mesh after each iteration. Sub-dividing the patches might lead to degenerate patches, which are points or lines. This might cause serious numerical problems later. We avoid this by always dividing the patch into four sub-patches at the center point.

The rectangular mesh, of course, presents problems of their own: it is harder to construct a closed smooth surface from the rectangular mesh because of the poles. But we can get around this problem, and then upgrade the closed rectangular mesh to C^1 closed surface.

Here, we currently use a Linear B-spline surface for its efficiency in computation time and some geometric properties we need. For a Linear B-spline surface, each control point only affects its four neighboring patches. This makes it very easy to separate the whole surface into independent strips.

4.2.3 The Powell minimization procedure

What we are performing now is simply the minimization of energy E_{total} by adjusting the points on the mesh. Gradient descent is not very reliable. For our application, an eligible numerical method should meet the following requirements:

1) It should be able to handle discontinuous functions,
2) it should work when the derivative information is unavailable,
3) it should be reliable and accurate,
4) the time complexity should be reasonable, and
5) convergence should be guaranteed.

Powell is a good candidate, though might be slower than gradient descent. But, in our application, it is not really the case because (1) we are using a coarse-to-fine approach and only the bad patches are handled at each iterations and (2) the weight W_{int} of smoothness constraint is changing implicitly as the process goes on. So, the gradient descent procedure, if applied, has to inverse a completely different matrix every time it is invoked even if the number of variable is the same (due to different W_{int}). It is impractical to pre-compute all inverse matrices ahead of time because there are infinite number of possible matrices. In contrast, Powell does not have to invert a matrix. Inverting a huge matrix, which might happen fairly often in our application, is time consuming. So, here the whole process should not be slowed down much by Powell.

The Powell algorithm is itself a direction set method for function minimization. Assume the function to be minimized has N variables. With an initial guess, which is an N-tuple vector, Powell can work by producing spontaneously mutually conjugate (non-interfering) directions, and searching along these directions sequentially for the (local) minimum. Because of the power of Powell, we can define almost any kind of energy we need. If we have any *a priori* about our target, we can define it in terms of the energy and apply it to Powell. For instance, if the approximate area AREA of the object is known, we can define an energy, $E_{area} = |AREA - AREA_{fitting\ surface}|$, using this information. According to our experiments, Powell is efficient, compared to the regular gradient descent approach, on the aspect of the number of control points needed to fit the data points. For more information, we refer the reader to [49,33,31,40,48,54].

If too many control points need to be handled simultaneously, Powell can become very slow. In our algorithm, this issue is addressed by a partitioning scheme, explained later, and thus Powell is always called with a limited number of variables. and we can get reasonable results in our experiments in a few minutes for simple objects.

4.2.4 Coarse-to-fine approach

Our approach is iterative, and we start this fitting process with a small number of control points (patches). At each iteration, we categorize the patches into two class-

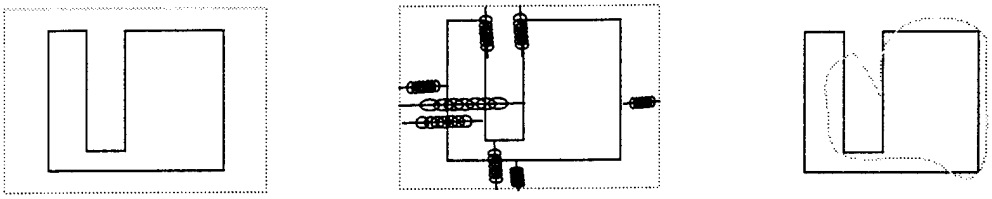
es, good and bad, based on their associated external energy (error fit). Only the control points belonging to the bad patches are adjusted to minimize the energy (error) of the fitting surface, and then all patches are subdivided into four sub-patches after the fitting process. In our current implementation, the user specify the number of the iterations. So, the process terminates after we have iterated a certain number of times. The numbers of the control points and patches increase as the iteration goes on. Presently, we iterate at most three times, which means that we only go through at most two subdivisions. The coarse-to-fine approach helps improve the performance in time very much. The global information can be acquired in the first 2 iterations, and the following iteration is used to highlight the surface details.

4.2.5 External energy E_{ext}

The external energy is a potential energy which attracts the fitting surface toward the data points. The result of the energy minimization highly depends on the external energy field. In our implementation, we define the external energy of a point on the fitting surface to be the distance to the nearest data point. The fitting surface can approach the data points when the external energy of the fitting surface is being minimized.

The external energy, which derives from the data points, can decide the success of the fitting process. A good definition of the external energy should has reasonable time and space complexities when the external energy of the fitting surface is being calculated. One possible solution to this is to define the external energy also from the data point's viewpoint. In addition to the energy field, we might define one more external energy based on the distance between each data point and the fitting curve. This might bring about another serious problem. How do we determine the corresponding point on the fitting curve for each data point? Usually, this information is not available. We might define the corresponding point to be the closest point on the fitting curve. But in reality the nearest points are not necessarily the corresponding points, and it is possible for one point on the fitting curve to correspond to more than one data point as shown in Figure 4.1 when the target is concave. In (a), the concave polygon, composed of solid lines, is the data points, and the dotted rectangle is the initial fitting curve. (b) shows possible wrong attachments based on the closest point criterion. Data points belonging to different parts of the target might be attached to the same part of the fitting curve, and, in consequence, the fitting curve could not capture the profile of the target faithfully as shown in (c). One more problem with this approach is that it is quite expensive, as we need to calculate the closest point for each data point every time the external energy is evaluated.

Our proposed method is efficient in the time and space complexities. We use an energy field represented by a cube composed of voxels. Each voxel contains the distance to nearest data point, and this distance is the energy of the voxel. The external energy of the fitting surface is calculated by sampling points on the surface and getting the energies of the voxels they fall in. This way, we can compute the external en-



(a) Data points and initial guess (b) Possible wrong attachments (c) Possible result

Figure 4.1 A possible mistake for the closest point approach

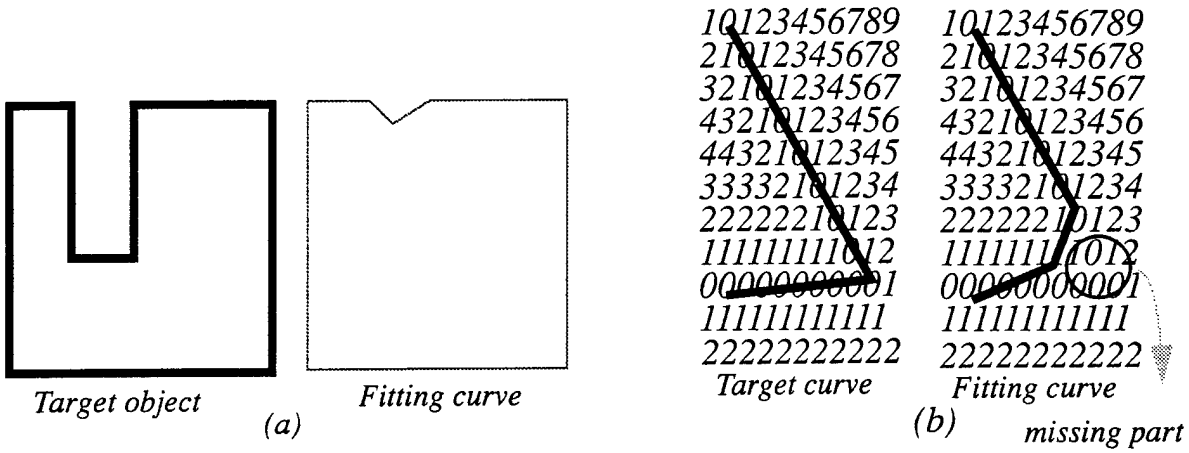


Figure 4.2 Problems with the long-distance external energy

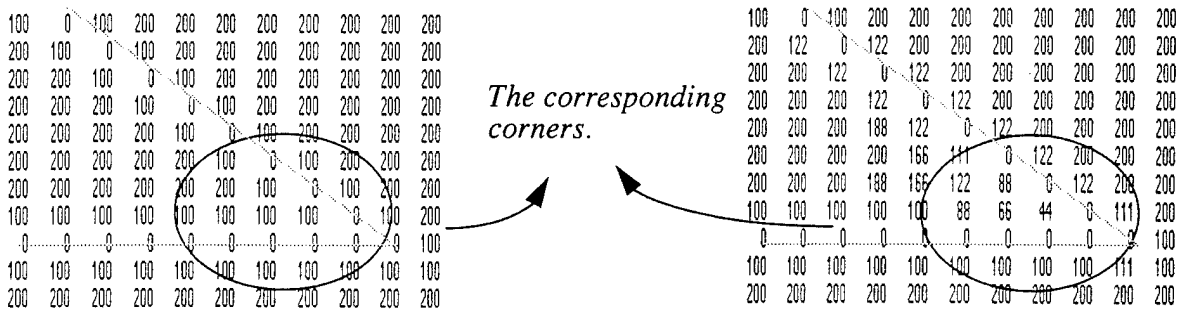
ergy at a low cost, and the space complexity, which is in proportion to the number of voxels in the cube, is reasonable.

The problems with this definition of distance, as the ones commonly encountered with morphology operations, is that it rounds corners and cannot handle cavities. These two problems are illustrated in Figure 4.2 below. In (a), the resultant curve only gives a dimple instead of going deep into the cavity. (b) shows the distance field and the target curve, and the resultant fit, which rounds the corner. The numbers in this example indicate the external energy at each point, and the pixel with energy zero is the data point. Any algorithm based on the shortest distance to the data points cannot be immune to these problems. To palliate this, we define two external energy fields, a long-distance one and a short-distance one. The long-distance external energy field is rather coarse, and measures the distance from each voxel to the nearest data point. Its main purpose is to quickly pull the fitting surface towards the data. The short-distance one is a more accurate measure.

At the beginning, we use the long-distance external energy, and when the fitting surface is close to the fitted object as measured by the long-distance external energy, we switch to the short-distance external energy to improve the results.

Long-distance external energy field:

The long-distance field is computed by a 3D Blum Medial Axis Transform [32] in a digitized cube $G(X,Y,Z)$ typically $150 \times 150 \times 150$. This algorithm is straightforward:



(a) the long-distance external energy. (b) the short-distance external energy.

Figure 4.3 An example of the long-distance external energy field and its corresponding short-distance external energy field

Every voxel is initialized to a very large number, except the data voxels which are zero. We then put a one in any voxel for which any of its neighbors contains a 0, and so on.

Short-distance external energy field:

Once the surface is close enough (the distance is smaller than H, typically 3), we redefine the energy field by averaging the values of the original field in an HxHxH neighborhood. Of course, we leave the 0 values (data) unchanged. This is equivalent to interpolating the original values. Figure 4.3 shows an example illustrating the difference between the long-distance and the short-distance external energies. It should be clear to the reader that the corner in the short-distance energy field is more salient than in the long-distance energy field.

Figure 4.4 shows an example of the effect of the short-distance external energy field. In this example, (a) is the initial data, which is a head, (b) is the fitting result applying the long-distance external energy field only, and (c) is the fitting result applying both the long-distance and the short-distance external energy field. (d), (e), and (f) are cross-sections of (a), (b), and (c), respectively. We can clearly see that we obtain a better fit if the short-distance external energy is applied. The use of short-distance external energy can also bring down the computational time to some degree.

We could combine these two external energies into one. The problem is that we use one byte to store the external energy of each voxel in our implementation, and the span of the energy value of the voxel for short-distance external energy field is much smaller, so we can get better quantization (resolution) for the short-distance external energy field, and thus obtain a better result. This is why we have two separate external energy fields here. We switch to the short-distance external energy field after the first iteration in our implementation.

In summary, the long-distance external energy field is coarse and brings the surface close to the data, and the refining work is left to the short-term external energy field.

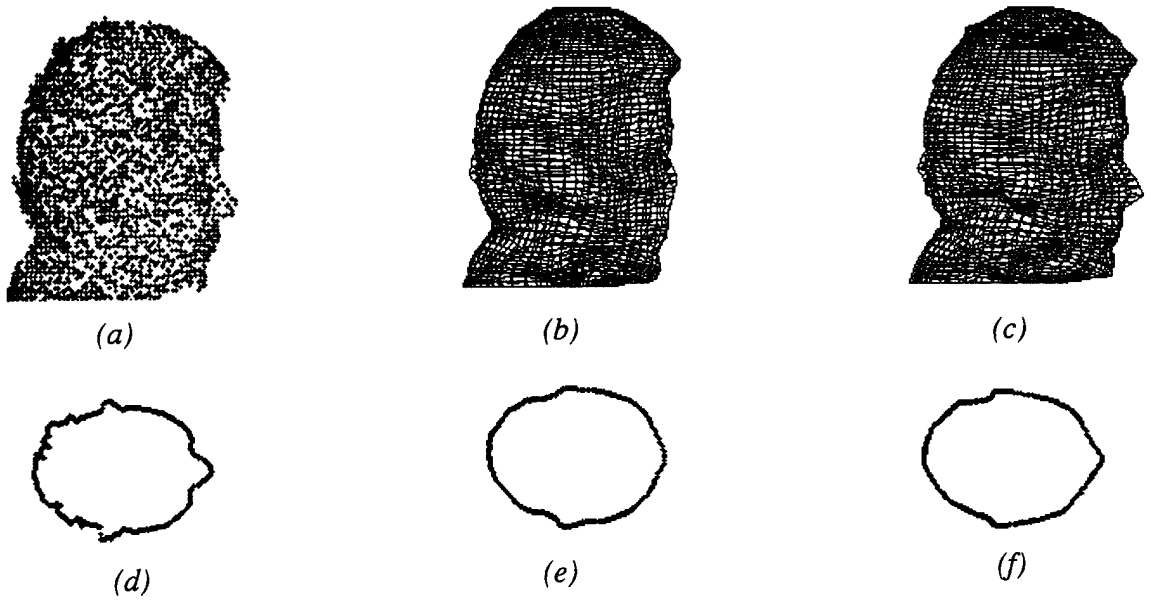


Figure 4.4 An example of the effect of the short-distance external energy

4.2.6 Internal energy E_{int}

The internal energy should be computed as a measure of the curvatures on the surface. Here instead, we measure it as the sum of the digital curvatures on some curves drawn on the surface. The choice of these curves will be explained later.

For a B-spline surface, parameterized by u and v, with MxN control points, we can construct M (resp. N) snakes, each of which contains N (resp. M) control points, directly from the control points along the u (resp. v) direction. We use the second derivative of these snake to represent the internal energy. Due to the high similarity between the shapes of the control points and the fitting surface, we can just directly use the control points to estimate the internal energy. The way we estimate the internal energy brings down the computational time since the internal energy is calculated very frequently. Suppose we have a curve composed of points P_i, $0 \le i < N$, the internal energy is defined as below:

$$E_{int} = \sum_{i=1}^{N-1} \left\| P_{i-1} + P_{i+1} - 2P_i \right\|^2$$

The reason why we define the internal energy in such a simple way, which has no arc length as the denominator, is that the length of the snake (curve) cannot change much when the fitting surface is close to the fitted object. We always place much more weight on the external energy than internal energy, so we don't expect the snake length to be influenced much by the internal energy. This formulation favors coplanar and equidistant point arrangements.

4.2.7 Choice of the initial surface

Now we move on to how the initial surface is set up. Currently, we have two methods for setting up the initial surface.

The first is to build a cylinder that covers all the data points as the initial surface. The second is based on a spherical coordinate representation. We compute the center of mass of the data first as the center of the initial sphere, then sample data points in $N_\theta \times N_\Phi$ directions, and find out the farthest data point in each direction. The radius in each direction is the distance between the center of mass and the corresponding farthest data point. We use this deformed sphere as an initial guess. If we do not find a data point in a given direction, we use the average of the radii in the neighboring sampled directions as the one in this direction.The caps of these two initial surfaces are constrained to be planar.

We always start with the second approach, which can give a better approximation of the data. If we are unable to compute the radius in many (more than 18 for 12×17 sampled directions) directions, we abort this choice and revert to the cylindrical initial guess.

4.3 Overview of our algorithm

4.3.1 Issues

Several problems come with the algorithms dealing with deformable model:

1) Huge computational time and space: Assuming M×N control points on the fitting surface, there are 3MN variables for this surface. Theoretically, we can just inject these 3MN variables into a minimization algorithm to minimize the energy of the fitting surface. This approach turns out to be impractical due to the unbearable computational time when 3MN is large. Furthermore, most minimization algorithms need a matrix of size (3MN)×(3MN), which results in huge space complexity also. An adaptive approach can just alleviate these problems, but cannot avoid these problems in the worst case, especially when all control points or patches are bad.

2) Constructing a smooth closed surface from the B-spline control points: This is no problem if the surface is constructed from the triangular mesh, but we choose rectangular mesh, and we have already given the reasons in the previous section. The problem is that the poles are not smooth if we try to construct the closed smooth surface, which is topologically equivalent to a sphere, directly from the control points. It is because these two poles are shared by many degenerate patches (triangular patches) around them.

3) A good approximation of the data as the initial surface: The quality of the fitting result depends on the initial surface, and we would like the result to be invariant to the initial surface as long as the initial surface is not too bad.

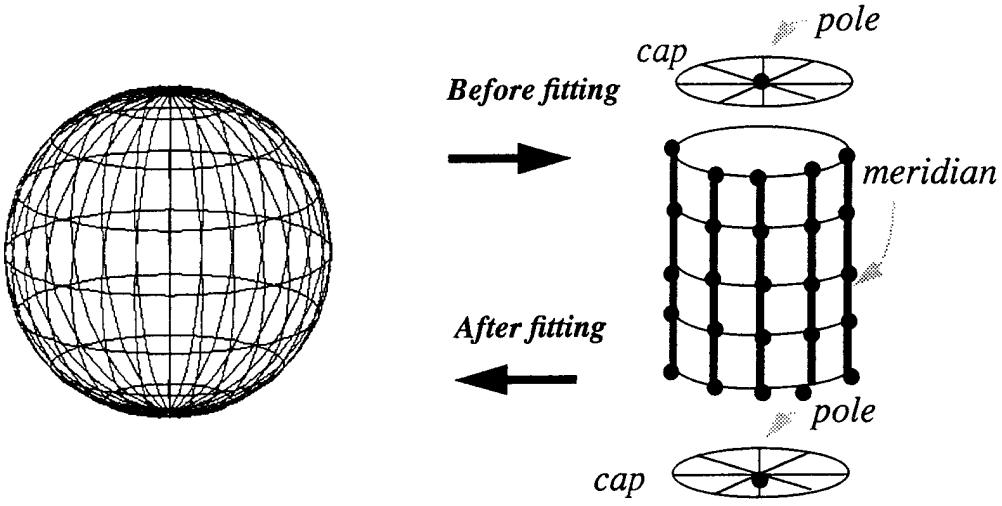


Figure 4.5 The initial closed surface and the definitions of Pole, Meridian, and Cap

4) The convergence of the surface to the data points: The convergence of the fitting process should be guaranteed.

Now, we define some terms for further use. We define a Cap to be the triangular patches formed by a Pole and its adjacent control points. So, we always have two Caps. A Meridian (a line of constant u in parameter space) is defined to be the line connecting the two Poles, as depicted in Figure 4.5 .

4.3.2 The algorithm

A flowchart of our algorithm is in Figure 4.6 .

Both the cylindrical and spherical initial surfaces we adopt are topologically equivalent to a sphere. In our algorithm, we consider a sphere as composed of three parts, which are two caps and an open cylinder as shown in Figure 4.5 . These three parts are processed separately in our algorithm.

Instead of injecting all MxN control points into the minimization procedure (which is possible but extremely expensive), we decompose the problem into a curve fitting problem followed by a (simpler) mesh fitting problem.

Given that the caps are already in place, we select every other meridian and move their (M-4) control points, which are not shared with the two caps, to minimize their energy. We then select the remaining meridians and move their (M-4) control points, which are not shared with the two caps, to minimize the energy of the related patches this time. It is important to note, however, that only the bad segments (patches or curves) are injected into the minimization procedure (after the energy minimization, we can guarantee that the associated energy of the bad segments is reduced, but some segments with higher external energy might still be bad). Then, we subdi-

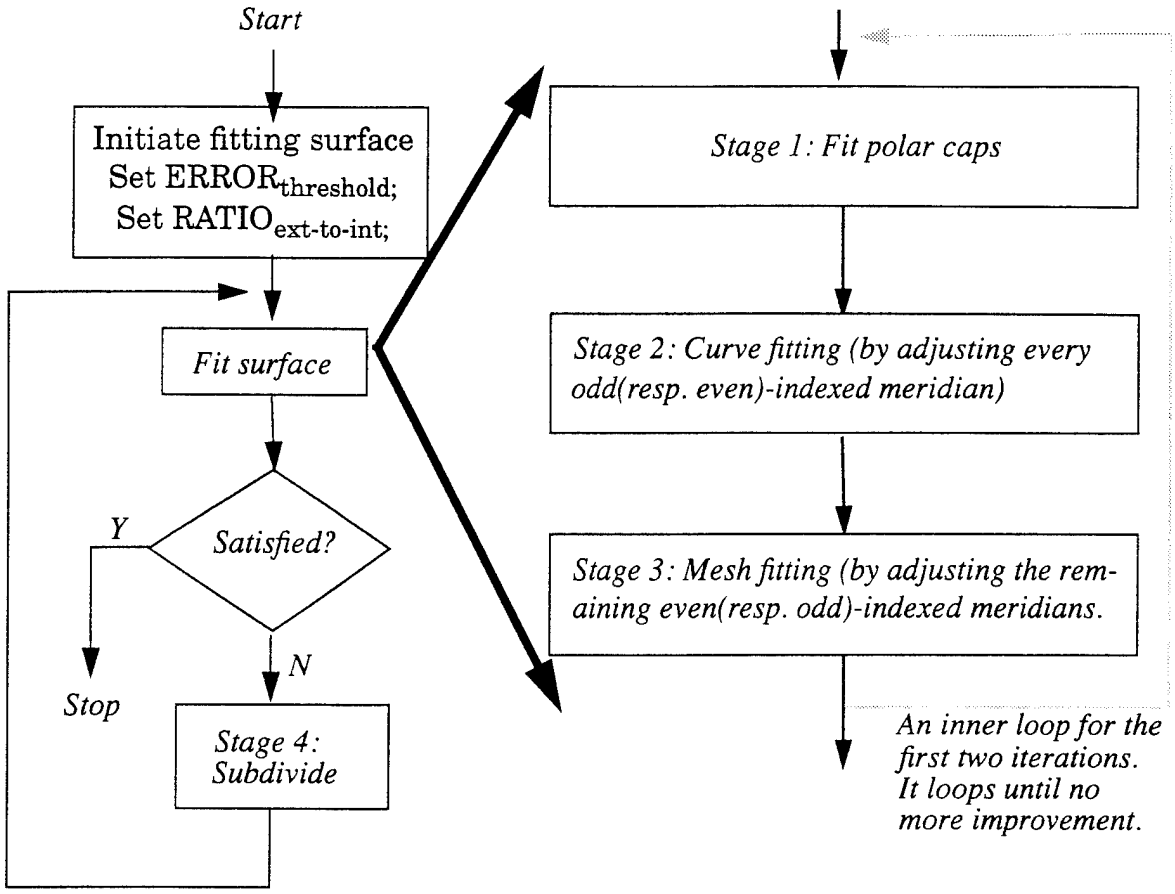


Figure 4.6 The flowchart of our algorithm

vide all patches in four, and repeat the process until some terminating condition is met.

Our algorithm is a 4-stage one, and during the first three stages, Powell is called frequently for energy minimization.

First, we fit the caps to the data, and force the caps to be planar. This way, all tangent vectors in all directions at the pole are coplanar. When fitting, the cap can change its shape, subject to the planar constraint, in order to get the best fit. Thanks to this planar constraint, the constructed surface is smooth even at the poles.

Second, we perform the curve fitting to some meridians to locate the profile of the target, and this is done by applying energy minimization to these meridians. We select the odd (resp. even)-indexed meridians and fit them to the data by treating them as approximating linear B-snake[44]. The only difference between a B-snake and a selected meridian lies in the internal energy. When calculating the internal energy of these meridians, we not only consider their own smoothness but also the smoothness between them and their immediate neighboring even (resp. odd)-indexed meridians (an example is depicted in Figure 4.8). Then we let them adapt to find the

profiles of the target. These selected meridians are not influenced much by the fitting surface (by the internal energy) when moving, so they can detect the object more accurately. Please notice that the external energy of these selected meridians is defined on the curve without considering the surface nearby. The surface is separated by the selected meridians into independent "strips" in a way, and each strip contains an even (resp. odd)-indexed meridian. Each strip is bounded by two odd (resp. even)-indexed meridians.

Next, we fit each strip to the target. We select meridians of the other polarity, even (resp. odd)-indexed, to do the mesh fitting for each strip. We treat them as regular snakes, except that they are tuned to minimize the external energy (error) of the strip. This means the external energy is not only from the curve but also from the area it defines.

The fourth stage is subdivision. If the fitting surface up to now is not satisfactory, we subdivide all rectangular patches into four and then go to stage 1; otherwise, exit.

4.3.3 Summary and discussion

In summary, in the first step, we fit the caps. Next, the odd (resp. even)-indexed meridians are used to find the profile or frame of the target, and the surface is divided into strips by these meridians. Then the even (resp. odd)-indexed meridians are applied to fit the strips to the data. Finally in stage four, we subdivide the surface, that is, we divide each rectangular patch into four. We break the 3D surface problem into a set of 2D (linear) B-snake problems in a way. Also notice that, at each step, although we have different ways of calculating the internal energy E_{int}, and the external energy E_{ext}, the basic idea is the same.

We can see that this is a typical coarse-to-fine approach. We start with few control points and large patches, then we increase the number of the patches and control points in later iterations.

Our algorithm overcomes the time and space complexities, and closed surface problems by (1) breaking the 3D surface problems into several 2D snake problems, which is shown in stages 2 and 3, (2) coarse-to-fine approach, and (3) forcing the cap to be planar, which is explained to stage 1. Due to the robustness of Powell, we do not need a good initial guess, and two examples are shown in Figure 4.17 andFigure 4.18 . Also, the Powell method guarantees convergence.

Please notice that stages 2 and 3 can be performed very fast when there are few control points. We can take advantage of this to get more reliable global information. We repeat these 2 stages until there is no further improvement in the first 2 iterations (in the first 2 iterations, there are not many control points in the meridians or caps). It is almost impossible to rectify the error from the first 2 iterations, which is considered global, by later iterations. So, in our implementation we add a inner loop to these two stages for the first two iterations.

The computational time could also be largely reduced by parallel processing. It is obvious that (1) the two caps are independent of each other, (2) all odd (resp. even)-indexed meridians are independent of one another, and (3) all even (resp. odd)-indexed meridians are independent of one another, so each stage can be performed in parallel.

There are two important parameters, $ERROR_{threshold}$ and $RATIO_{ext\text{-}to\text{-}int}$, set by the user in this algorithm.

$ERROR_{threshold}$ is used to determine whether or not a patch of the surface or a span of the snake is good. We only process the bad parts of the snakes and the bad patches on the fitting surface during each iteration. At each iteration, a patch (or a span) is good if its average external energy is smaller than $ERROR_{threshold}$; otherwise, it is bad.

$RATIO_{ext\text{-}to\text{-}int}$ specifies the relative importance of the external energy with respect to the internal energy. After setting $RATIO_{ext\text{-}to\text{-}int}$, two internal parameters W_{ext} and W_{int}, concerning the weights of the external and internal energies, are set by the system. W_{ext} is always 1, and W_{int} is set as below:

$$W_{int} = \frac{E_{current-ext}}{E_{current-int} \times RATIO_{ext-to-int}}$$

where $E_{current\text{-}ext}$ is the current external energy of the fitting surface or snake, anda

$E_{current\text{-}int}$ is the current internal energy of the fitting surface or snake.

Every time Powell is invoked, W_{int} is re-calculated based on $RATIO_{ext\text{-}to\text{-}int}$ and the current internal and external energies of the fitting surface. So every time, Powell may be called with different W_{int}.

The reasons why we set RATIOext-to-int is that Wext and Wint are different measures and thus on different scales. $RATIO_{ext\text{-}to\text{-}int}$ serves to normalize two energies. $RATIO_{ext\text{-}to\text{-}int}$ is always greater than 1; otherwise, the fitting surface is unlikely to conform to the data points, as we now explain.

As we subdivide the surface after each iteration, the fitting surface is approaching the real object. We have more confidence in the fitting surface. So it is suggested that the internal energy weight be reduced as the process goes on. One more advantage of $RATIO_{ext\text{-}to\text{-}int}$ is that the weight of the internal energy tends to decrease as the process goes on, because E_{ext} decreases faster than E_{int} does when $RATIO_{ext\text{-}to\text{-}int}$ is greater than 1. By setting $RATIO_{ext\text{-}to\text{-}int}$ to be a constant greater than 1, we can diminish the importance of the internal energy after each iteration implicitly, and thus obtain a better fitting surface.

On the contrary, if we set W_{int} directly and keep it unchanged, then the internal energy tends to dominate at the later iterations because the external energy decreases faster than the internal energy does. This might not lead to a good fit.

$ERROR_{threshold}$ and $RATIO_{ext\text{-}to\text{-}int}$ are 1.0 and 10 in all of our experiments.

4.4 Details of our algorithm

Now, we elaborate on the four stages concerning the cap, curve and mesh fittings, and subdivision.

Stage 1. Cap fitting

By treating the pole as 2 or more control points, we can achieve C^0 continuity at the poles on the caps when constructing the fitting surface. For example, for quadratic B-spline surface, we can achieve C^0 simply by duplicating the control points at the two poles. The caps of the surface present problems when we try to upgrade the surface from C^0 to C^1, because the poles will not be C^1 and are singular. This is an inherent limitation of the rectangular mesh no matter what surface construction algorithms (for example, B-spline and Bezier) are employed. But if the control points are coplanar, then the tangent vectors at the poles along all directions will be coplanar. Based on this, we constrain all the points, including the pole and its adjacent control points to be coplanar. This way the surface constructed can be smooth everywhere.

The cap can be represented by the following formula. which contains (N+5) variables, where N is the number of the adjacent points of the pole. Initially, φ and ψ are

$$(x_i, y_i, z_i) = \left(R_i^2 \cos\theta_i, R_i^2 \sin\theta_i, 0\right) \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \times \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} + (x_p, y_p, z_p) \quad ,0 \le i < N.$$

where (x_p,y_p,z_p) is the coordinate of the pole,

N is the number of points adjacent to the pole,

(x_i,y_i,z_i) are the coordinates of the ith point adjacent to the pole,

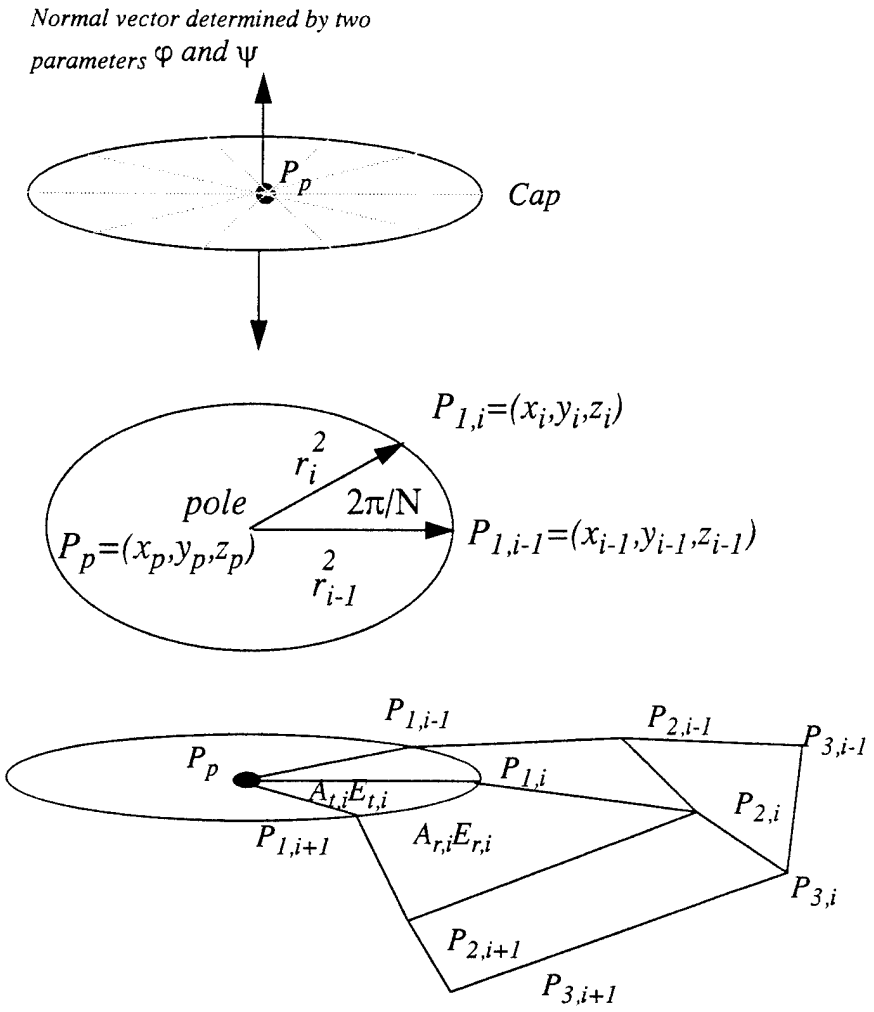
$q_i=2p*i/N$,

R^2_i is the distance from (x_p,y_p,z_p) to (x_i,y_i,z_i),

φ is the angle of the rotation around the x axis, and

ψ is the angle of the rotation around the z axis.

zero. The distance from the pole to an adjacent point should be always positive, so we use the square root of the real distance as the variable to guarantee that the distance is positive. The variables in this formula are x_p, y_p, z_p, φ, ψ, and R_i ,$0 \le i < N$, so the number of variables is N+5. Initially, these variables can be set in accordance to the initial



$P_{1,i}$ and $P_{1,i-1}$ are two consecutive control points on the row immediately adjacent to the pole.

Figure 4.7 An example of the cap

closed surface (the caps of the initial surface should be planar surely). An example is depicted in Figure 4.7 .

The internal energy E_{int} for the cap is as below:

Let $P_{1,i}$ and $P_{2,i}$, $0 \le i < N$, be the two closest rows to the pole ($P_{1,i}$ is the row immediately adjacent to the pole.).

$$E_{int} = \left(\sum_{i=0}^{N-1} \|P_p + P_{2,i} - 2P_{1,i}\|^2 + \|P_{1,i} + P_{3,i} - 2P_{2,i}\|^2 + \|P_{1,i-1} + P_{1,i} - 2P_{1,i+1}\|^2 \right)$$

The external energy for the cap is:

$$E_{ext} = \sum_{i=0}^{N-1} (A_{t,i} \cdot E_{t,i} + A_{r,i} \cdot E_{r,i})$$

Where $A_{t,i}$ is the area of triangle $P_pP_{1,i}P_{1,i+1}$,

$E_{t,i}$ is the average external energy of triangle $P_pP_{1,i}P_{1,i+1}$,

$A_{r,i}$ is the approximate area of rectangular mesh $P_{1,i}P_{1,i+1}P_{2,i+1}P_{2,i}$, and

$E_{r,i}$ is the average external energy of rectangular mesh $P_{1,i}P_{1,i+1}P_{2,i+1}P_{2,i}$.

The area factor is introduced in the above formula because the contribution of each patch should be in proportion to its size.

$E_{t,i}$ and $E_{r,i}$ can be estimated by

1) sampling a certain number of points in the patch,
2) finding the energy of each point from array G defined in the previous section, and finally
3) averaging the external energy.

$A_{r,i}$ can be coarsely estimated by dividing the rectangular patch into four triangles through the central point, and then summing the area of every triangle.

Every time before Powell is called, the weights W_{int} and W_{ext} are determined by $RATIO_{ext\text{-}to\text{-}int}$, the current internal energy E_{int}, and external energy E_{ext} of the cap. Then, Powell is called to minimize $E_{total} = (W_{int}*E_{int} + W_{ext}*E_{ext})$ of the cap by adjusting those (N+5) variables. There are two independent caps on the surface, so this step will be executed twice.

Stage 2. Curve fitting

Suppose there are MxN control points on the surface, that is, there are M rows and N columns. Here, N is always even. Let N=2K. There are N Meridians on this surface.

Now we select the Meridians with odd (or even) index to do the curve fitting (and the rest are for the mesh fitting, explained later), so there are K such meridians.

At the beginning of this stage, these selected odd(resp. even)-indexed meridians are the same as the corresponding meridians at the previous iteration (or, for the first iteration, the corresponding meridians of the initial sphere) except for those control points shared with the caps.

We treat each odd(resp. even)-indexed meridian individually as if they are independent of one another. Each odd(resp. even)-indexed meridian is just like an ordi-

nary snake except that it is also influenced by the control points on the two neighboring Meridians.

The first two and last two control points of the odd(resp. even)-indexed meridian are shared with the caps, so these four points are supposed to be good and won't be tuned under any situation. Each meridian has M control points and (M-1) spans. Because the first and the last spans are fixed by the caps, we only need to handle at most (M-3) spans for each odd(resp. even)-indexed meridian when doing curve fitting.

A odd(resp. even)-indexed meridian is tuned adaptively, and we just deal with the bad spans of the meridian at each iteration. We check the external energy of each span. Those spans with average external energy greater than $ERROR_{threshold}$ are bad. We find out the bad connected spans (bad connected spans have to be processed simultaneously). This way, we can classify those bad spans into several groups, and each group is independent of each others. Let G_s be one of the groups (G_s is itself a miniature snake in some sense). An example is shown in Figure 4.8 . In this example, let the solid thick curve, composed of P_{21}, P_{31}, and P_{41}, be G_s. The dotted and the solid line segments show the control points involved when calculating the internal energy. The external and internal energies are as below. In this example, there are 3 control points P_{21}, P_{31}, and P_{41} to adjust, which leads to 9 variables injected to Powell.

The average external energy can be obtained by sampling a certain number of

$$E_{int} = \left(\sum_{i=1}^{5} \left\| P_{i+1,1} + P_{i-1,1} - 2P_{i,1} \right\|^2 + \sum_{i=2}^{4} \left\| P_{i,0} + P_{i,2} - 2P_{i,1} \right\|^2 \right)$$

$$E_{ext} = \sum_{i=1}^{4} \left\| \overline{P_{i,1} P_{i+1,1}} \right\| \cdot E_i$$

where E_i is the average external energy of the line segment between $P_{i,1}$ and $P_{i+1,1}$.

points, finding the external energy of each sampled point through array G, and then averaging those external energies. The length of each line segment is considered because the importance of each line segment should be in proportion to its length.

The weights of those two energies are determined in the same way as we do for the cap in the previous stage.

Suppose there are H_s control points in G_s, excluding those shared with the caps.

The coordinates of each control point are the three components X, Y, and Z, so we have $3*H_s$ variables.

For all G_s's on all odd(resp. even)-indexed meridians, we call Powell to minimize

E_{total} (= $W_{int}*E_{int}$ + $W_{ext}*E_{ext}$)

of G_s by adjusting the corresponding $3*H_s$ variables.

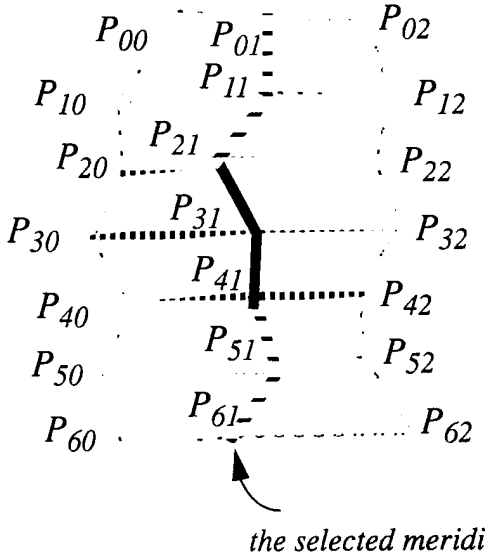


Figure 4.8 The odd(resp. even)-indexed meridian

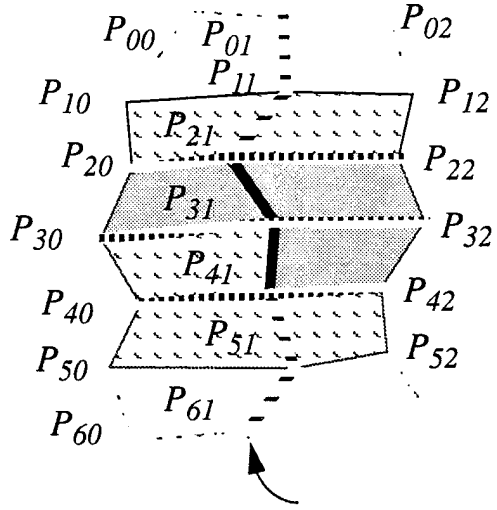


Figure 4.9 The even(resp. odd)-indexed meridian.

We may say that the responsibility of odd(resp. even)-indexed meridians is to find the frame or profile of the object.

Stage 3. Mesh fitting

The surface now is separated by caps and odd(resp. even)-indexed meridians into K independent strips, and each strip can be tuned independently now. We select the rest even(resp. odd)-indexed meridians, and use them to fit all strips to the data (target). These even(resp. odd)-indexed meridians are fixed in stage 2 for the curve fitting. Each strip contains an even(resp. odd)-indexed meridian.

At the beginning of this stage, the even(resp. odd)-indexed meridians are the same as the corresponding meridians at the previous iteration (or, for the first iteration, the corresponding meridians of the initial sphere) except for those control points shared with the caps.

The responsibility of the selected meridians in this stage is to fit each strip to the data (target) by mesh fitting (We assume that the odd(resp. even)-indexed meridians have done a good job in stage 2 here).

For each strip, there are 2*(M-1) patches and an even(resp. odd)-indexed meridian. Because those four patches shared with the caps are supposed to be good by default, we only need to consider 2*(M-3) patches. Let S be one of those strips. Now we need to determine the goodness of those 2*(M-3) patches in S. A patch is bad if its average external energy is greater than $ERROR_{threshold}$. We find the connected bad patches (these bad patches need to be processed simultaneously), and this way the bad patches in strip S can be classified into several groups.

Let G_m be one of those groups, and S_m be the intersection of G_m and the even(resp. odd)-indexed meridian in the strip. S_m is itself a small snake, and the internal energy of G_m is calculated through S_m. An example is in Figure 4.9 . In this example, let the 3 dotted rectangular patches be G_m, and the thick solid line segments composed of P_{21}, P_{31}, and P_{41}, be S_m. The dotted and the solid line segments show the control points involved when calculating the internal energy (13 control points are involved), and the diagonal patches together with the dotted ones are those patches involved when we calculate the external energy of the even(resp. odd)-indexed meridian (there are 8 patches involved). In this example, the control points tuned are P_{21}, P_{31}, and P_{41}, and thus there are 9 variables injected to Powell for this example.

The internal and external energies for the example in Figure 4.9 . are as below:

$$E_{int} = \left(\sum_{i=1}^{5} \left\| P_{i+1,1} + P_{i-1,1} - 2P_{i,1} \right\|^2 + \sum_{i=2}^{4} \left\| P_{i,0} + P_{i,2} - 2P_{i,1} \right\|^2 \right)$$

$$E_{ext} = \sum_{i=1}^{4} \sum_{j=0}^{1} A_{i,j} \cdot E_{i,j}$$

Where $\overline{E}_{i,j}$ is the average external energy of rectangular mesh $P_{i,,j}P_{i,,j+1}P_{i+1,,j+1}P_{i,,j+1}$, and $A_{i,j}$ is the approximate area of the rectangular mesh.

The way to compute the internal energy is exactly the same as that for odd(resp. even)-indexed meridian in stage 2, and the way to estimate the area of a patch and the average external energy is explained in the previous section for the cap.

Suppose S_m contains H_m control points, and then there are $3*H_m$ variables to be tuned by Powell.

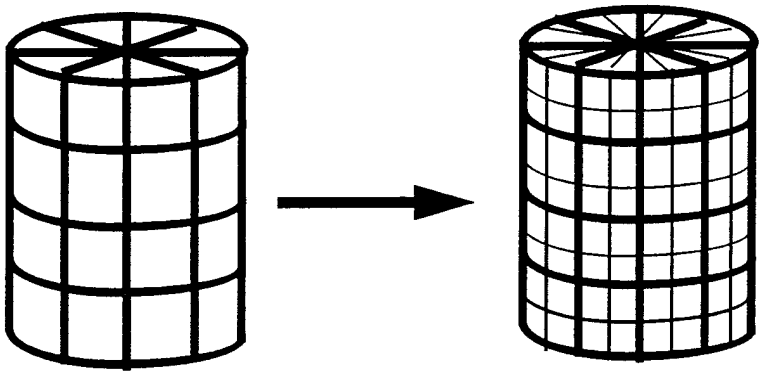


Figure 4.10 Subdivision. Notice that no row is added on the cap

For all G_m's on all even(resp. odd)-indexed meridians, we call Powell to minimize

E_{total} (= W_{int}*E_{int} + W_{ext}*E_{ext})

of G_m by adjusting the corresponding 3*H_m variables.

Stage 4: Sub-division

At this step, every rectangular patch is divided into four, except those on the caps. We do not add a new row to the cap because these triangles around the pole are small already, so adding a new row to the cap might lead to degenerate triangles. This is illustrated in Figure 4.10 .

These four stages in the flowchart correspond to those in the previous paragraph. If the cap is already close to the real object, that is, it has low external energy, then we can skip this stage. The following two stages are to do the curve and mesh fittings. We have an inner loop for these two stages in the first two iterations. In this inner loop, we choose the odd-indexed and even-indexed meridians in turns in these two stages for the sake of fairness. The fourth stage is to do the subdivision. We switch from the long-distance energy field to the short one after the first iteration.

4.5 Experiments

We now show results from eight experiments on real data. Five of them are on the tooth, wood, phone, head1 (Carol), and head2, respectively. One shows how the fitting surface of the head1 experiment evolves, and the last two show the robustness of this algorithm by giving a very bad initial surface. Then we show one more experimental result on synthetic data, which shows the difference in the surface normal between the fitting surface and the underlying synthetic surface.

In all of these eight experiments, we start with the same crucial parameters to show the robustness and stability of this algorithm. $ERROR_{threshold}$ is 1.0. $RATIO_{ext\text{-}to\text{-}int}$ is 10.

These experiments are performed on a Sun Sparc-10 workstation. For each experiment, we show the original data and two shaded results. The shaded surface is

based on the C^1 B-spline surface, which is constructed directly from the control points obtained (in fact, these control points are for C^0 B-spline surface). All surfaces shown here are C^1 and closed, including the two poles.

The general information of the first five experiments is listed in Table 3: and Table 3:Because the cap of the surface is represented by a set of parameters rather than control points, the numbers of control pointsare approximated numbers..

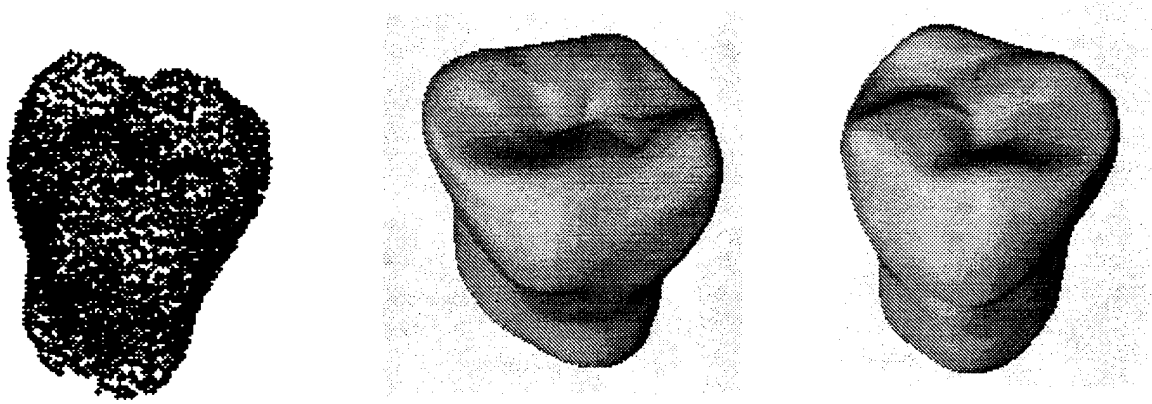
Table 4: Performance Summary (cont.)

	size of the external energy cube	initial average error in voxels (cylinder initially)	final average error in voxels (cylinder initially)
tooth	150X150X150	4.42	0.6
wood	64X64X64	3.0	0.27
phone	150X150X150	5.62	0.39
head1-Carol	300X300X300	18.00	0.78
head2	300X300X300	20.15	1.18

Table 3: Performance Summary

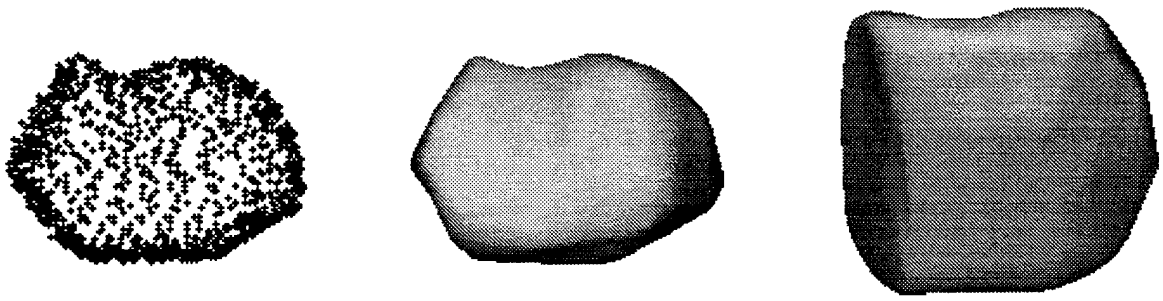
	number of data points	number of control points	run time (cylinder initially)	run time (sphere initially)	number of subdivisions
tooth	11841	263	4 mins	3 mins	1
wood	5562	212	1.5 mins	1.0 mins	1
phone	15776	247	4.5 mins	N/A	1
head1-Carol	136082	2632	21.5 mins	14 mins	2
head2	45514	2632	31.5 mins	24 mins	2

In the tooth experiment, we start with 11841 data points. A 150x150x150 cube is used to store the external energy. The running time is around 4 minutes. There are around 263 control points used on the resultant surface. The result is in Figure 4.11 . The initial surface is a cylinder in this experiment, we also tried the spherical initial surface, and it does not make any significant difference, because the object itself is not very complex. The surface is subdivided once, which means there are two iterations. One of the poles is on the top in this experiment, and we can see the top of the tooth is smooth (the other pole is at the bottom). The average external energy of the surface point is initially 4.42 voxels, 1.0 voxel after the first iteration, and 0.6 voxel after the second iteration (one sub-division has been done).

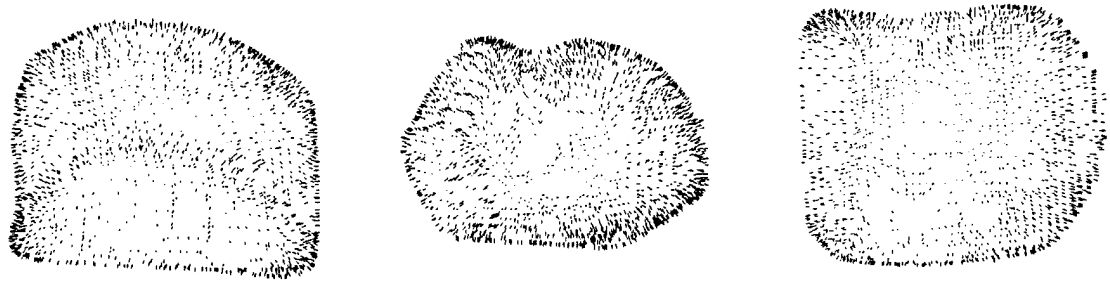


(a) data points *(b) shaded result 1* *(c) shaded result 2*

Figure 4.11 Tooth



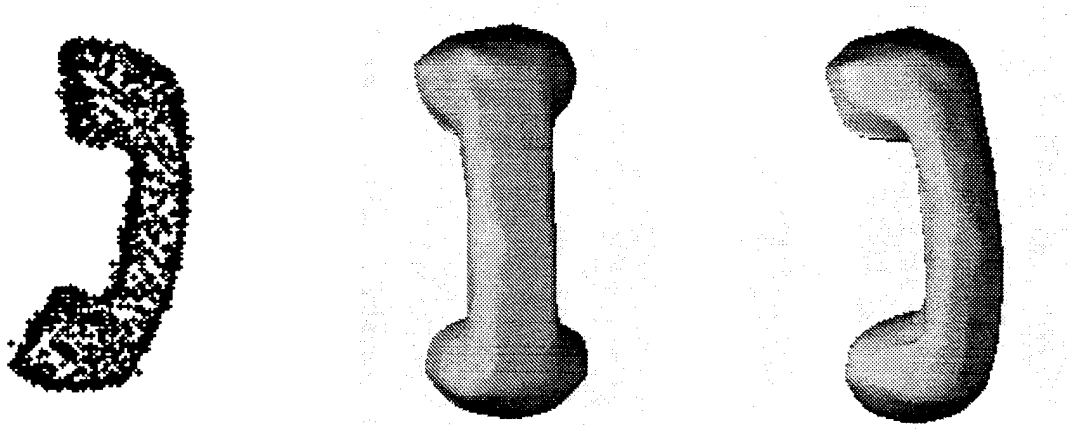
(a) data points *(b) shaded result 1* *(c) shaded result 2*



(d) Three views of the distribution of the error in pixel

Figure 4.12 Wood block

In the wood experiment, we start with 5562 data points. A 64x64x64 cube is used to store the external energy. The running time is around 1.5 minutes. There are around 212 control points used on the resultant surface. The result is in Figure 4.12 . The initial surface is a cylinder in this experiment, we also try the other algorithm for the initial surface, and it dose not make significant difference because the object itself is not too complex. The surface is subdivided once, which means there are two iterations. The two poles are on the left and right sides, respectively. The average external energy of the surface point is initially 3.0 voxels, 0.41 voxel after the first iteration,



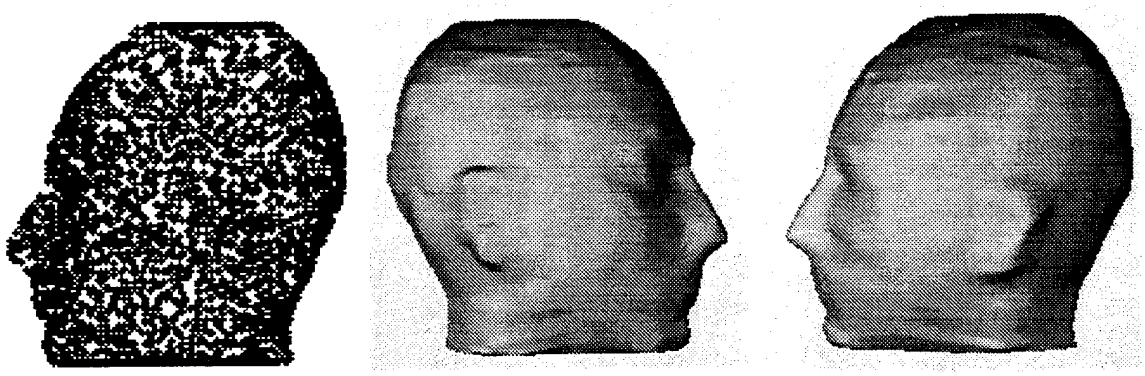
(a) data points *(b) shaded result 1* *(c) shaded result 2*

Figure 4.13 Phone

and 0.27 voxel after the second iteration (one sub-division has been done). The distribution of the external energy is shown in (c). The length of each vector in (c) is in proportion to the external energy of the associated surface point.

In the phone experiment, we start with 15776 data points. A 150x150x150 cube is used to store the external energy. The running time is around 4.5 minutes. Around 247 control points are used for the resultant surface. The result is in Figure 4.13 . The initial surface is a cylinder. The other initial surface construction algorithm cannot be applied because the center of mass falls outside the object and we cannot sample any points in many directions, so the system switches to the cylinder scheme for the initial surface. The surface is subdivided once, which means there are two iterations. The two poles are at the top and the bottom, respectively. The average external energy of the surface point is initially 5.62 voxels, 0.45 voxel after the first iteration, and 0.39 voxel after the second iteration (one sub-division has been done).

In the head1 experiment, there are 136082 data points. A 300x300x300 cube is used to store the external energy. This set of data is from INRIA, by courtesy of professor Ayache. We project the data points onto 3D space and fill up the top and bottom of the head, which are hollow originally. We consider points in the same voxel one point, so the number of data points here is different from that of the original one.We also tried both initial surface algorithms. The running time is around 21.5 minutes for the cylindrically initial surface, and 14 minutes for the spherical one. Around 2632 control points are used on the resultant surface. The results from these two different initial surfaces are much alike, so we just show the result in Figure 4.14 from the cylindrical initial surface, which is harder. Figure 4.14 a shows the data points, and Figure 4.14 b and Figure 4.14 c are the results. The computational time is reduced by spherical initial surface because there is already some geometrical information in it. The surface has been subdivided twice, which means there are three iterations. The two poles are at the top and the bottom, respectively. The average external energy of the surface point is initially 18.00 voxels, 1.45 voxels after the first iteration, 0.88 vox-



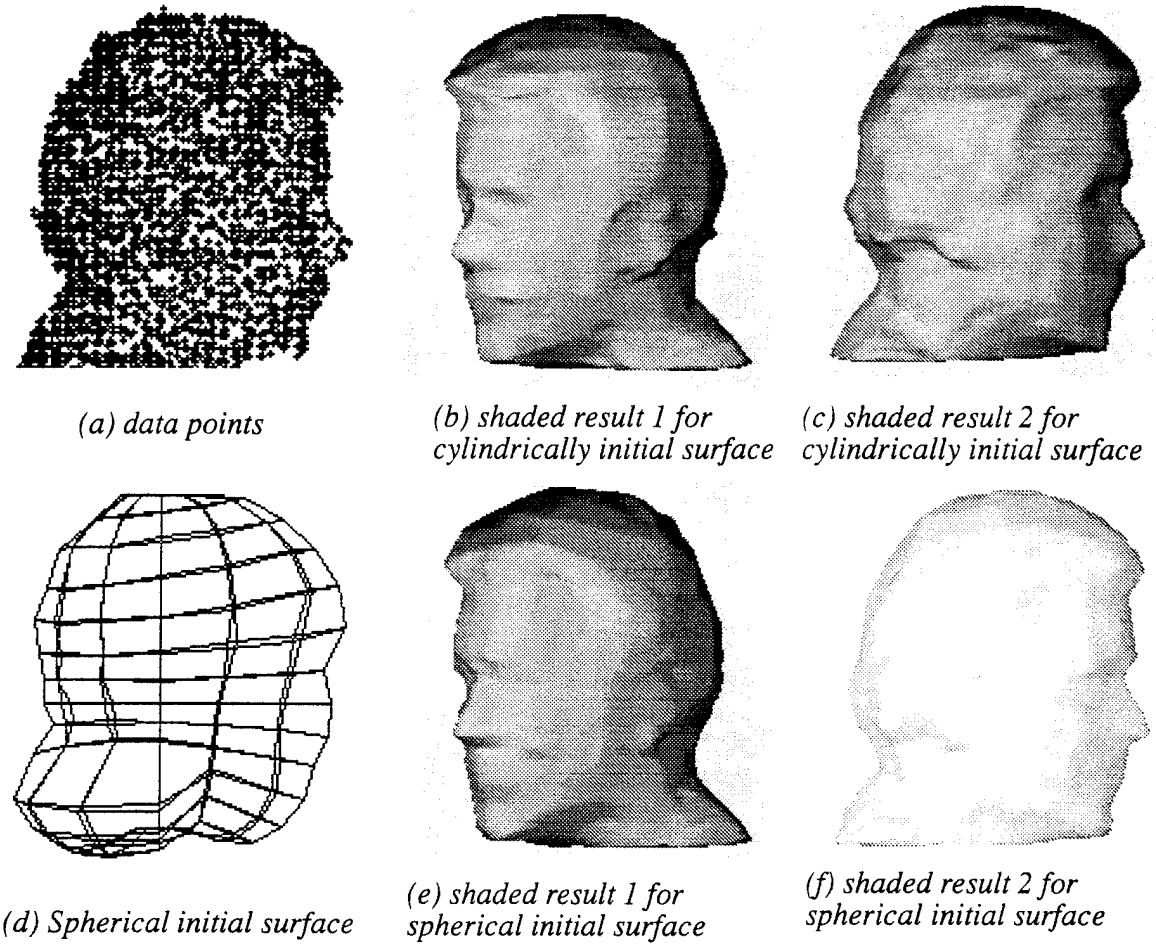
(a) data points *(b) shaded result 1* *(c) shaded result 2*

Figure 4.14 Head1 (Carol).

el after the second iteration (one sub-division has been done), and 0.78 voxel after the third iteration (two sub-divisions have been done). It is interesting to note that our number of vertices (2632) is significantly lower than the one reported by others. In Gueziec's [36] experiment on this data, 256x128(=32768) control points are used, and in Nastar's [46] experiment, 11130 nodes are used. One more difference between our result and theirs is we form a closed C^1 surface with the top and bottom closed.

In the head2 experiment, we start with 45514 data points. A 300x300x300 cube is used to store the external energy. This set of data is from the Media Lab, MIT. For the same reasons as in the head1 experiment, the number of data points here is different from that of the original one. We tried both algorithms for the initial surface. The running time is around 31.5 minutes for the cylindrically initial surface, and 24 minutes for the spherical one. There are around 2632 control points used on the resultant surface. Figure 4.15 shows the result. In Figure 4.15 , (a) shows the data points, and (b) and (c) are the results from the cylindrically initial surface, (d) shows the initial spherical wire-frame surface, and (e) and (f) show the result from the spherical initial surface. Figure 4.16 shows the evolution of the fitting surface with the cylindrically initial surface, and both the shaded and wire-frame results are shown. The average external energy of the surface point is initially 20.15 voxels, 1.43 voxels after the first iteration, 1.21 voxels after the second iteration (one sub-division has been done), and 1.18 voxels after the third iteration (two sub-divisions have been done). In this experiment, we can tell that the result from the spherical initial surface is a little bit better, and can be obtained faster. It is because the spherical initial surface has already captured some geometrical properties, and thus simplifies the calculation to some degree. The surface is subdivided twice, which means there are three iterations.

We also performed an experiment to try the robustness of this algorithm by giving a bad spherical initial surface. The data points here are the same as those of head2 experiment. Suppose there are two directions in which we cannot sample and data point, and the interpolation scheme fails, then there are two deep cavities on the initial surface as shown in Figure 4.17 . In (a), we shows the wire frame of the bad spher-



(a) data points

(b) shaded result 1 for cylindrically initial surface

(c) shaded result 2 for cylindrically initial surface

(d) Spherical initial surface

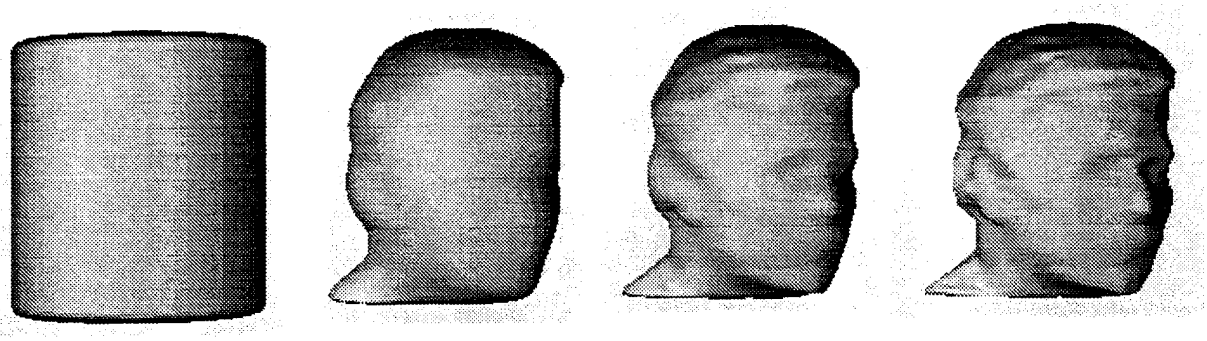
(e) shaded result 1 for spherical initial surface

(f) shaded result 2 for spherical initial surface

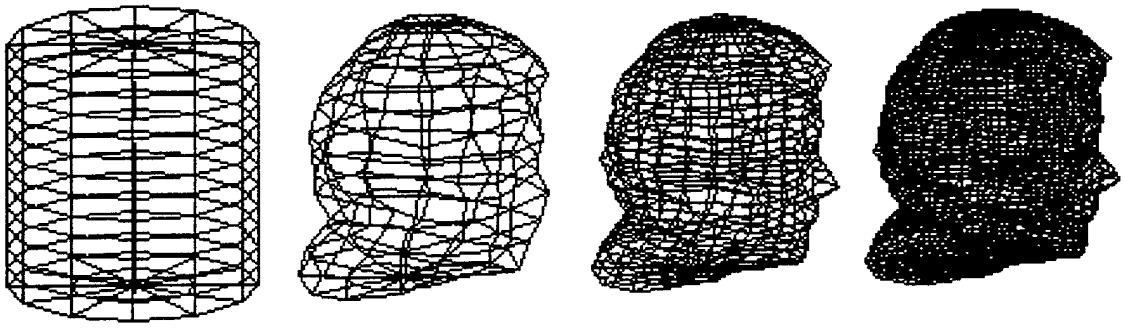
Figure 4.15 Head2

ical initial surface with two cavities at the top of the head and jaw. (b) and (c) show the cavities at the jaw and the top, respectively. (d) is the shaded result after the energy minimization. (e) and (f) show the counterparts of the cavities in (b) and (c), respectively.The cavities have been filled, and the final surface in (d) is fine. In (e) and (f), we can also see the poles are well handled, and the poles on the constructed surface, which are at the top and bottom, are very smooth. The computation is around 24 minutes for the experiment. Based on our experiments on data points head2, this algorithm can tolerate up to 18 cavities, and we obtain similar results in around 24 minutes. Figure 4.18 shows the experiment on the initial surface with 18 deep cavities. (a) shows the defective initial surface, and (b) shows the corresponding views of the final surface. It is evident that all the cavities have been handled. This indicates strong resistance against the bad initial surface.

From the above experiments, we can see that there is not much difference in the quality for these two different initial surface schemes, and this algorithm can also tolerate, to some extent, the error in the initial surface. The main difference is the com-



(a) Initial surface. *(b) result 1* *(c) result 2* *(d) result 3*



(e) Initial wire frame *(f) Wire frame 1* *(g) Wire frame 2* *(h) Wire frame 3*

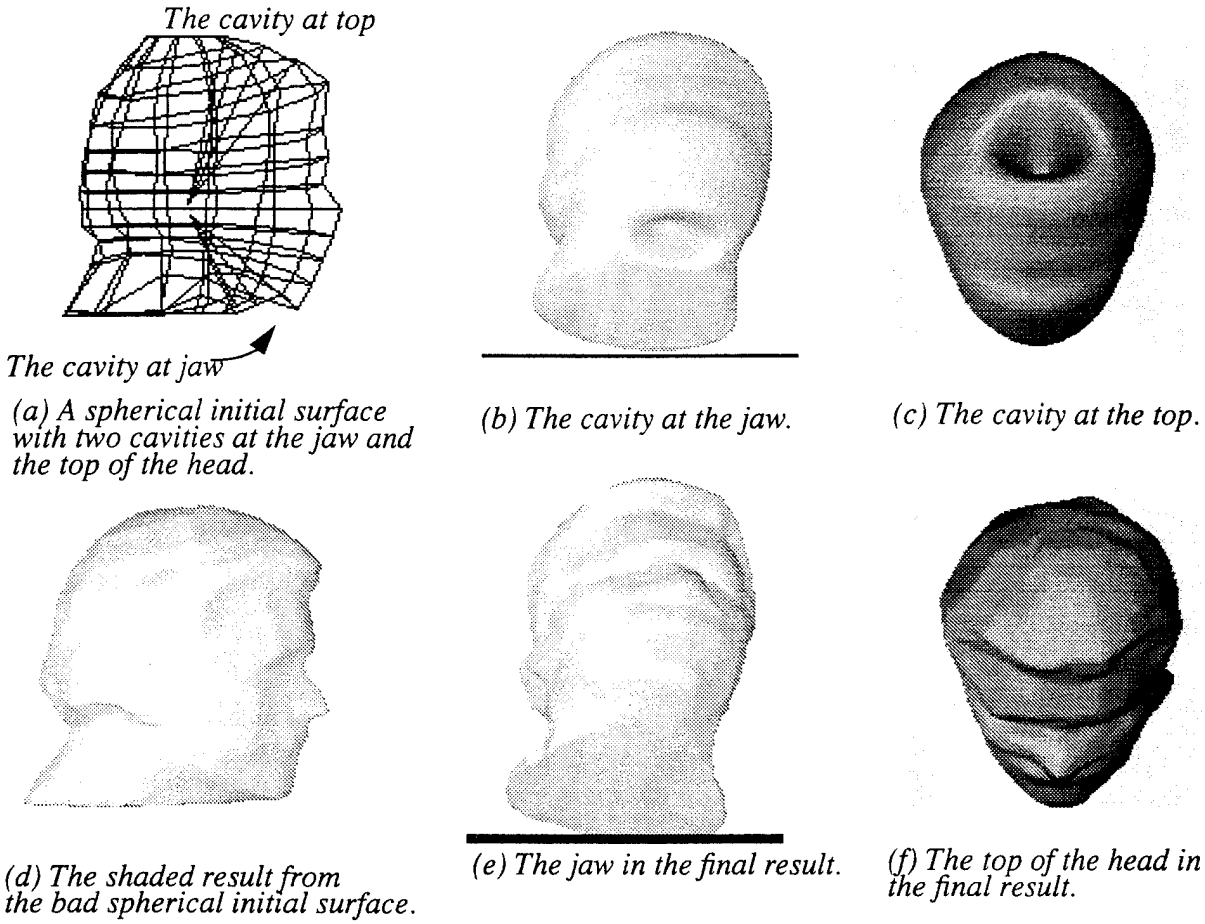
Figure 4.16 The evolution of the experiment of head 1. (a) is the initial surface (cylinder). (b), (c), and (d) are the deformed results for each iteration. The surface has been sub-divided twice. (e), (f), (g), and (h) show the wire frames of (a), (b), (c), and (d).

(a) One view of the defective initial surface.

(b) The corresponding view of (a) of the final surface.

Figure 4.18 A test of the robustness on the defective surface with 18 deep cavities

putation time especially when the fitted object is complex. This shows the robustness of this algorithm because it works well when good initial surface is unavailable at the expense of longer computational time.



(a) A spherical initial surface with two cavities at the jaw and the top of the head.

(b) The cavity at the jaw.

(c) The cavity at the top.

(d) The shaded result from the bad spherical initial surface.

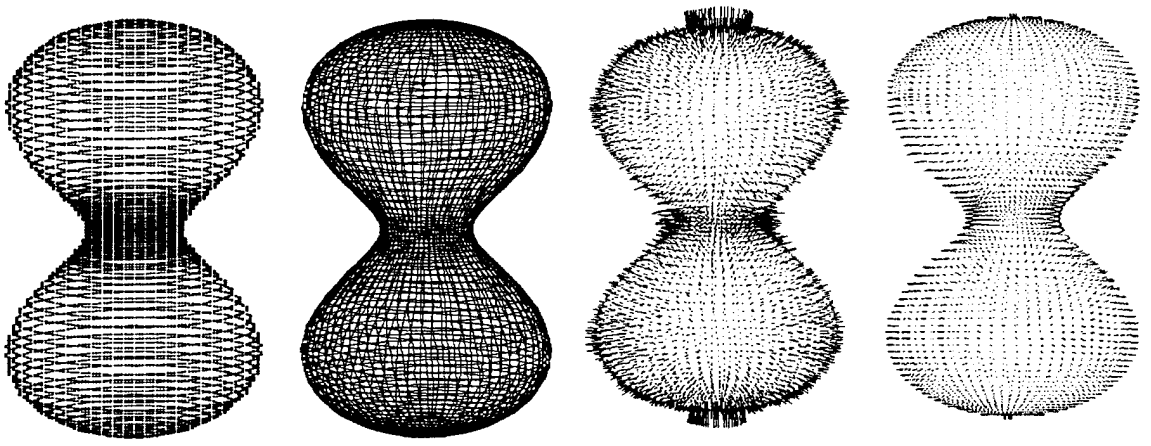
(e) The jaw in the final result.

(f) The top of the head in the final result.

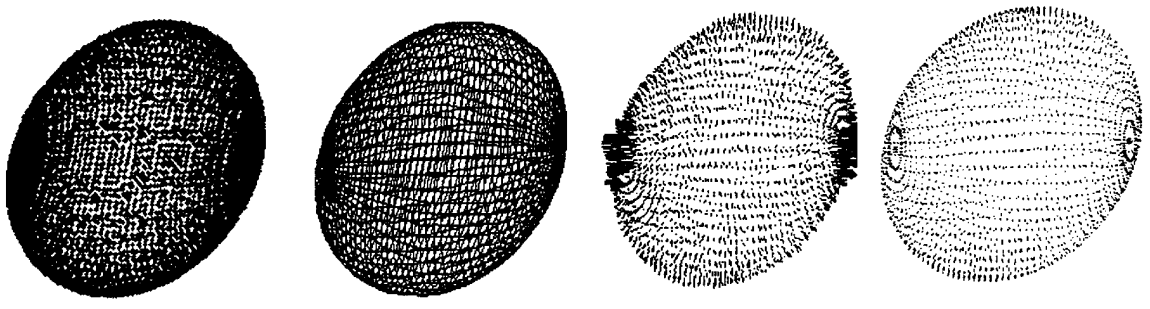
Figure 4.17 A test of the robustness of this algorithm

In head1 and head2 experiments, there are discontinuities at the bottom (for both experiments) and at the top (for head1 experiment). They are flat at those parts. From the results shown so far, we can see those discontinuities come out well because we reduce the importance of the smoothness constraint as the process goes on, and we can handle discontinuities.

In Figure 4.19 we show the difference in surface normal between the fitting surface and the underlying synthetic surface. We conduct this experiment because the normal and the shape of the surface are highly correlated. The normals of the fitting surface and the underlying object might be different while they are physically very close to each other. This experiment is to show how faithful our algorithm is in terms of the normal. Synthetic data are used so we can calculate the normal of the underlying object. It turns out after this experiment that the difference in the normal is rea-



(a) Raw data *(b) Fitting result* *(c) Error in normal* *(d) External energy*



(e) Raw data *(f) Fitting result* *(g) Error in normal* *(h) External energy*

Figure 4.19 Estimates of the error in terms of the normal.

sonable using our approach.We use the following two implicit functions to generate the synthetic data points:

$(x^2+y^2+(z-a)^2)\cdot(x^2+y^2+(z+a)^2) = b^4$ where a=67.68, and b=68.96.

and

$\left(\frac{x}{a}\right)^2+\left(\frac{y}{b}\right)^2+\left(\frac{z}{c}\right)^2 = 1$ where a=80, b=40, and c=60.

First we sampled points from these two functions as shown in (a) and (c). A 200x200x200 cube is used to store the external energy. (b) and (f) are the fitting results, respectively. (c) and (g) show difference in the surface normal between the fitting surface and the real surface. (d) and (h) show the distribution of the external energy, respectively, and the length of the vector reflect the magnitude of the external energy at this point. The average external energies of (d) and (h) are 0.224 voxels and 0.5 voxels. The length of each vector in (c) and (g) reflects the difference between the two corresponding normals of a surface point. These vectors are exaggerated a little bit so we can see the distribution of the error easier. Please notice high error points concentrate at the poles. It is because we enforce the planar constraint at the polar

areas which leads to these artifacts. In the other places, the errors are reasonable. In

Table 5: Errors

	size of the external energy cube	final average error in voxels	[0,3)	[3, 6)	[6,9)	[9,1 2)	[12,1 5]	>1 5	total
peanut	200X200X200	0.224	4239	310	25	85	0	0	4659
ellipse	200X200X200	0.5	1926	143	54	47	5	0	2175

(c) 4659 surface points are sampled. 4239 points have an error between 0° and 3°, 310 points between 3° and 6°, 25 points between 6° and 9°, 85 points between 9° and 12°, and no points have an error more than 12°. In (g) 2175 surface points are sampled. 1926 points have an error between 0° and 3°, 143 points between 3° and 6°, 54 points between 6° and 9°, 47 points between 9° and 12°, 5 points between 12° and 15°, and no points have an error more than 15°. This information is also shown in Table 5:.

4.6 Discussion

There are several important aspects in this paper:

1. We do not use the traditional adaptive approach for this application because it is almost impossible to determine whether a patch is good or not correctly. Our new scheme is a coarse-to-fine approach. It divides all patches after each iteration. It is still efficient because if a patch is really good, then the only operation applied to it in the future is just sub-division, which costs very little. This scheme also preserve the rectangular structure of the surface after each sub-division, which makes generating smooth surface easier and cheaper. This approach is free from the degenerate patch problem because a rectangular patch is always divided into 4 rectangular ones.

We prefer the rectangular mesh to the triangular mesh because it is much easier to construct a smoother surface from the rectangular mesh, and the properties, such as derivatives, are much easier to obtain.

2. We use linear B-splines. On the one hand, it is the cheapest, and on the other hand, it might divide the surface into independent strips. If higher degree B-splines are employed, the relationship between the control points and the patches is more complex.

3. There is always a large matrix associated with the minimization algorithm, and the size of the matrix is in proportional to the square of the number of the variables. This might result in the memory explosion if there are many control points to handle at a time. Also, the numerical method goes extremely slow under this situation. We break a 3 dimensional problem down into several 2 dimensional problems,

and then the space and time complexities can be reduced significantly. We separate the surface into several strips, so Powell is always called with a limited number of variables. For example, if the fitting surface has MxN control points, the maximum number of variables sent to Powell is around 3*(N-4). Only the bad parts of the strips and the meridians are tuned by Powell. So, in practice, the number of variables is far below 3*(N-4). The caps only have (N+5) variables, which is also low.

4. We reduce the weight of the internal energy implicitly as the iteration goes on, because we have more confidence in the fitting surface after each iteration. This way, the discontinuities of the data can be well preserved.

5. When dealing with an open surface, which is much easier, we just skip the first stage (for the cap).

6. Powell can handle many kinds of functions. This gives us flexibility to define the energy for any specific property. Powell may also be replaced by other numerical methods. Compared to gradient descent, Powell is more accurate and reliable, and in our application Powell is not necessarily slower than gradient descent. From our experiments, we know our algorithm has significant tolerance against a bad initial surface. We attribute this stability and tolerance to minimization algorithm.

7. Through the coarse-to-fine approach, both the global structure and the detailed information on the fine parts of the object can be acquired at different iterations. The global information, which might be applied to recognition and database search, is obtained in the first iteration, and the details of the object can be obtained in the later iterations.

8. Due to the independency among the caps and meridians, our algorithm could run in parallel, so the computational time could be further reduced significantly by parallel processing.

9. This system is easy to control because there are only two global parameters to adjust. In all of our experiments, the same values were used.

10. Overall, this system is reasonable in robustness, accuracy, time complexity, and space complexity. Its output surface can be easily taken by the other surface generating algorithm to construct a smooth surface.

There are also some problems left:

1. We cannot handle the object with deep cavities, especially the through ones. This problem is alleviated by the short-distance external energy, but it is not solved yet. Futhermore, we also cannot differentiate more than two objects when they are very close to one another, and we might mistake them for one object. We are developing some algorithms in 2D to tackle these problems, and we believe they can be extended to 3D in the future.

2. We need a more systematical way to determine the number of the iterations needed. Now the number of iterations is specified by the user. In our experiment, we iterate three times for the complex objects and twice for the simpler objects in our experiments. According to our experiment, reasonable results can be come by in three iterations. Yet, we would like the system to differentiate the complex objects from simple ones by itself, so it can determine the number of iterations.

3. We also would like to find a way to set reasonable $ERROR_{threshold}$ and $RATIO_{ext\text{-}to\text{-}int}$ directly by the computer. Now, they are set by the user. Under some situation, we may not know much about the input data, so we would like our system to be able to set these two parameters reasonably under this situation.

4. The self-intersection of the surface is a potential problem, even though we have not observed it in practice. Theoretically, this can be solved by adding an energy term $E_{intersection}$, which is zero when there is no self-intersection, and infinite when the self-intersection occurs. We can rule out this possibility of the self-intersection if the there is no self-intersection on the initial surface, because Powell can guarantee that the function value of the result is always less than or equal to the that of the initial guess. The problem with this approach is it is too expensive. It is expensive to check if there is a self-intersection on the surface.

4.7 References

[31] Acton, Forman S. 1970, in *Numerical Methods That Work* (New York: Harper and Row), pp. 464-467.

[32] Blum, H. "A Transformation for Extracting New Descriptors of Shape," *Proceedings of Symposium on Models for Perception of speech and Visual Form*, W. Whaten-Dunn (ed.), MIT press, Cambridge, Massachusetts, 1967.

[33] Brent, Richard P. 1973, in *Algorithms for Minimization without Derivatives* (Englewood Cliffs, N.J.: Prentice-Hall), Chapter 7.

[34] Isaac Cohen, Laurent D. Cohen, and Nicholas Ayache, "Introducing Deformable Surfaces to Segment 3D images and infer differential structures," *Proceedings of Computer Vision and Pattern Recognition 1991,* pp.738-739, Hawaii, 1991.

[35] H. Delingette, M. Hebert, K. Ikeuchi, "Shape Representation and Image Segmentation Using Deformable Surfaces," *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.467-472, Maui, HI, June 1991.

[36] Andre Gueziec "Large Deformable Splines, Crest Lines and Matching," *Proceedings of International Conference on Computer Vision*,1993, pp.650-657, Berlin, Germany, May, 1993.

[37] Song Han, Dmitry B. Goldgof, and Kevin W. Bowyer "Using Hyperquadrics for Shape Recovery from Range Data," *Proceedings of International Conference on Computer Vision* 1993, pp.492-496, Berlin, Germany, May, 1993.

[38] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Wemer Stuetzle "Surface Reconstruction from Unorganized Points," *Computer Graphics*, 26, 2, July 1992, pp.71-78.

[39] W. C. Huang and D. B. Goldgof "ADaptive-Size Meshes for Rigid and Nonrigid Shape Analysis and Synthesis," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 15, No. 6, June 1993.

[40] Jacob, David A.H., ed. 1977, in *The State of the Art in Numerical Analysis* (London: Academic Press), pp.259-262.

[41] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models" , in *International Journal of Computer Vision,* January 1988, pp.321-331.

[42] C. W. Liao, and G. Medioni "Representation of Range Data with B-spline Surface Patches," *Proceedings of International Conference on Pattern Recognition 1992,* pp.745-748, Hague, Netherland, August, 1992.

[43] Tim McInerney and Demetri Terzopoulos "A Finite Element Model for 3D Shape Reconstruction and Nonrigid Motion Tracking," *Proceedings of International Conference on Computer Vision* 1993, pp.518-523, Berlin, Germany, May, 1993.

[44] Sylvie Menet, Philippe Saint-Marc, and Gérard Medioni, "B-snakes: implementation and application to stereo," in *Proceedings of Image Understanding Workshop 1990*, pp.720-726, Pittsburgh, September, 1990.

[45] Shigeru Muraki "Volumetric Shape Description of Range Data using 'Blooby Model'" *Computer Graphics,* Volume 25, Number 4, July 1991, pp.227-235.

[46] Chahab Nastar and Nicholas Ayache "Fast Segmentation, Tracking, and Analysis of Deformable Objects," technical report, No. 1783, INRIA, France

[47] Alex Pentland, and Stan Sclaroff, "Closed-Form Solutions for Physically Based Shape Modeling and Recognition" *IEEE trans. on Pattern Analysis and Machine Intelligence,* Vol. 13, No.7, July 1991, pp.715-729.

[48] Polak, E. 1971, in *Computational Methods in Optimization* (New York: Academic Press).

[49] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, in *Numerical Recipes in C, The Art of Scientific Computing* (Cambridge), Chapter 10.

[50] Francis J.M. Schmitt, Brian A. Barsky, and Wen-Hui Du "An Adaptive Subdivision Method for Surface-Fitting from Sampled Data" *ACM SIGGRAPH* 86, pp.179-188.

[51] Sarvajit S. Sinha and Brian G. Schunck "Surface Approximation using Weighted Splines," *Proceedings of Computer Vision and Pattern Recognition 1991*, pp.44-49, Hawaii, 1991.

[52] Sarvajit S. Sinha and Brian G. Schunck "Discontinuity Preserving Surface Reconstruction," *Proceedings of Computer Vision and Pattern Recognition 1991*, pp.229-234, Hawaii, 1991.

[53] Franc Solina and Ruzena Bajcsy "Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.12, NO.2, February 1990, pp.131-147.

[54] Stoer, J., and Bulirsch, R. 1980, in *Introduction to Numerical Analysis* (New York: Springer-Verlag).

[55] D. Terzopoulos, and D. Metaxas, "Dynamic 3D Models with Local and Global Deformations: Deformable Superquadratics," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 13, No. 7, July 1991, pp. 703-714.

[56] Demetri Terzopoulos and Manuela Vasilescu "Sampling and Reconstruction with Adaptive Meshes," *Proceedings of Computer Vision and Pattern Recognition 1991*, pp.70-75, Hawaii, 1991.

[57] .D. Terzopoulos, A. Witkin, and M. Kass, "Constraints on deformable models: Recovering 3D Shape and Nonrigid Motion," *Artifical Intelligence,* Vol. 36, 1988, pp. 91-123.

[58] M. Vasilescu and Demetri Terzopoulos. "Adaptive Meshes and Shells: Irregular Triangulation, Discontinuities, and Heretical Subdivision," *Proceedings of Computer Vision and Pattern Recognition 1992* pp. 829-832. Urbana-Champaign, IL, June, 1992.

5 Recovering Surfaces, 3-D Intersections, and 3-D Junctions Using Perceptual Constraints

Gideon Guy and Gérard Medioni

We treat the problem of recovering surfaces from an incomplete set of input measurements, by applying perceptual constraints to the data. This extends our 2D perceptual Grouping work [64] to three dimensions. We show how both the Extension fields and the Saliency indicators, which were used for the 2-D case, can be elegantly generalized to 3-D.

We treat sparse "clouds" of non-oriented points, oriented points, and partial surfaces, in a uniform and *non-iterative* way. We are able to handle scenes of any genus, any number of discontinuities, and of any number of objects, without a priori knowledge or special considerations. The result is in the form of three dense saliency maps for surfaces, intersections between surfaces, and 3-D junctions. These saliency maps can then be used to guide a "following" process to generate a CAD model of surfaces, space curves, and 3-D junctions. We present some preliminary results on computer-generated images.

5.1 Introduction

Perceptual organization has gained popularity in the Computer Vision research in the past few years, and its importance has been widely recognized. First proposed by Lowe [67], and later by numerous researchers (e.g. [59,60,69], and see [63] for a more complete review), perceptual considerations have been used for a variety of problems in Computer Vision. All of above attempts took as input two dimensional image features.

Among the perceptual constraints used, the most common are: Co-Linearity, Proximity, Simplicity, and Co-Curvilinearity. These constraints are used to handle gaps and errors in input data, and assist in a higher-level description. The same kind of task is present in 3-D inputs, where some 3-D data is available, but it is not complete and/or it is noisy. Such input data is normally acquired by range imaging or as a result of a process that finds depth from X (stereo, shape from shading etc.). Here the task is to describe the underlying surfaces.

Much work has been done in fitting surfaces to clouds of points. The deformable models approach (first proposed by Kass *et al.* [65] for 2D, and in [70] for 3D) attempts to deform an initial shape so that it fits a set of points through energy minimization. The deformable shape is subject to certain constraints (such as smoothness, stiffness

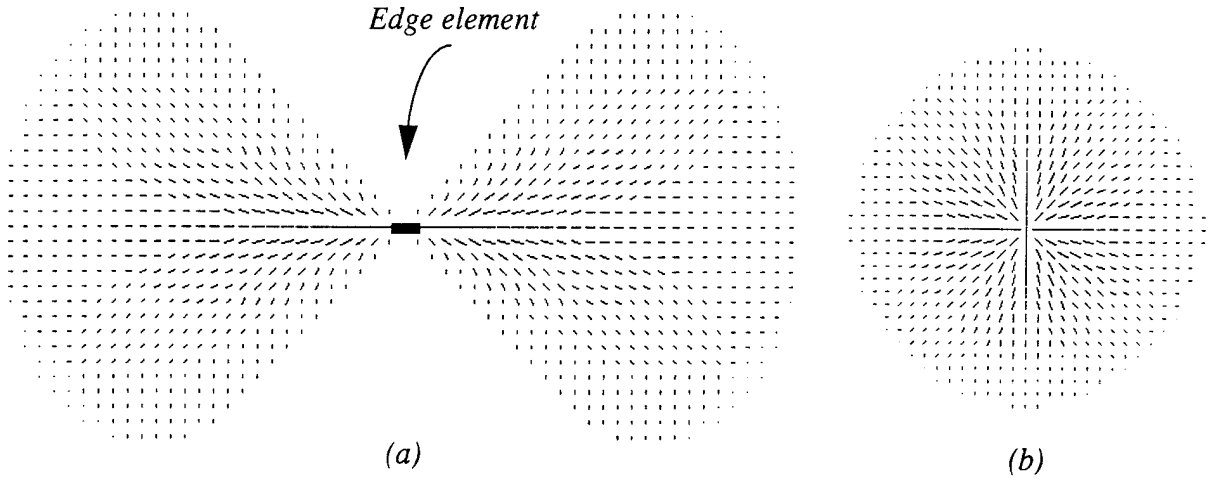


Figure 5.1 (a) The 2-D Extension field (b)the 2-D point field.

etc.) and the point data set behaves as an attractive energy field. Sander [61] have recently proposed a local algorithm to describe surfaces from a set of points. A quadratic surface is estimated around each data point, and an iterative process refines the local surfaces, and creates graph-like connections between compatible patches. Others have used similar methods (for a summary see [66]).

All of the above methods are computationally expensive as an iterative process takes place. Also in many of the methods, only one genus-zero object can be described at any one time, and surface boundaries and discontinuities are not always easy to describe.

We start by briefly discussing our older 2-D work, emphasizing the derivation and justification of the combination mechanisms. This will serve as an introduction to the derivation of the 3-D combination mechanisms, which share the same paradigms.

5.2 From 2D fields to 3D fields

In our 2-D work [64] we describe two basic fields, namely, the Extension field and the point field. The Extension field is used when edgels are present in the input, and the point field is used whenever non-oriented features are present. The 2-D version of the two fields are shown in Figure 5.1 The elements of the 2-D Extension field describe the most likely orientation of a curve passing everywhere in space. The above field is used as a mask in a so-called directional convolution, in which the mask is oriented along segments of the input image. votes that consist of strength and orientation are accumulated at each site of the image, to later determine the saliency of such site. The result is thus a saliency map, where high values denote high likelihood of a curve passing there. The actual combination at each site is described next and paves the way to the 3D case.

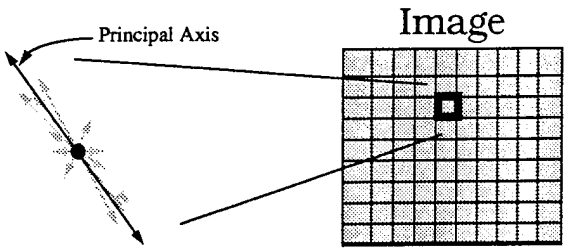


Figure 5.2 The principal axis of the votes collected at a site is taken as an approximation of the preferred direction.

5.2.1 Combination per site - The 2-D case

Ideally, we would want an averaged majority vote regarding the preferred orientation at a given position. In practice, we treat the contributions to a site as being vector weights, and compute moments of the resulting system. Such a physical model behaves in the desired way, giving both the preferred direction and some measure of the agreement. We use the direction of the principal axis ($EVmin$) of that physical model as the chosen orientation (See equation (5.1)).

$$\begin{bmatrix} m_{20} & m_{11} \\ m_{11} & m_{02} \end{bmatrix} = \begin{bmatrix} EVmin \\ EVmax \end{bmatrix} \begin{bmatrix} \lambda min & 0 \\ 0 & \lambda max \end{bmatrix} \begin{bmatrix} EVmin^T & EVmax^T \end{bmatrix} \tag{5.1}$$

This acts as an approximation to the desired majority vote, without the need to consider the individual votes.

The saliency map *strength* values are taken as the values of the corresponding λmax at each site. So, large values would indicate that a curve is likely to pass through this point. This map can be further enhanced (as shown in the next section) by considering the eccentricity, or $1 - (\lambda min/\lambda max)$. When that value is multiplied by the previous saliency map we achieve better selectivity, and only *curves* are highlighted. This results in a map defined by $\lambda max - \lambda min$.

5.2.1.1 Justification - The 2-D case

Basically, what we are looking for is a function that takes positive vectors as input and results in a measure of the agreement in their orientation. The result should satisfy several criteria:

- We want the result to be normalizable, so that we can compare different sites on a standard scale.
- The measure needs to be monotonically increasing with the addition of positive contributions.
- It should give higher values to 'better' (more directed) spatial arrangements of vectors.
- We want the effect of proximity to be independent of the affect of agreement.

It is easy to show how the model behaves when a single vector is added to it. Assume the variance-covariance matrix is as follows at state *t:*

$$C^t = \begin{bmatrix} m_{20}^t & m_{11}^t \\ m_{11}^t & m_{02}^t \end{bmatrix} \qquad (5.2)$$

The sum of the eigenvalues is the trace of the matrix:

$$\lambda_{\min}^t + \lambda_{\max}^t = m_{20} + m_{02} \qquad (5.3)$$

Now adding a new vector $V = [R\cos\theta, R\sin\theta]^T$ to the system will result in a new state *t+1*:

$$\lambda_{\min}^{t+1} + \lambda_{\max}^{t+1} = m_{20} + m_{02} + (R\cos\theta)^2 + (R\sin\theta)^2 = m_{20}^t + m_{02}^t + R^2 \qquad (5.4)$$

Note that the angle θ has disappeared on the r.h.s. of (5.4). This means that the sum of eigenvalues is independent of the orientations of the voting vectors and can hence be used as an indicator of proximity (a wider sense of proximity of course), and as a primitive saliency measure.

Equation (5.4) can obviously be written as:

$$\lambda_{\min} + \lambda_{\max} = \sum_{i=1}^{N} R_i^2 \qquad (5.5)$$

Where N is the number of segments in the original image.

We define the eccentricity $E = 1 - \lambda_{\min}/\lambda_{\max}$ as a measure of agreement. Obviously this value is between 0 and 1[1]. Our intuitive notion of 'agreement', or of a majority vote on a continuous scale, is consistent with the above definition. This means that in all cases where we feel that collection *A* has better 'agreement' than collection *B*, the corresponding eccentricity values will share the same relationship (i.e. *E(A)>E(B)*). This is not to say that both functions are equal, but merely that both are monotonic.

Eccentricity values by themselves cannot perform as saliency measures since sites with very little voting strength can produce high eccentricity values. In fact, consider a site far away from where the 'action' is, which accepts exactly *one* vote (This can happen in practice). The eccentricity value is 1, but the site is of no importance.

---

1. Since $\lambda_{\min} \leq \lambda_{\max}$ and are both non-negative for a semi-positive definite matrix.

However, Consider λ_{max} itself. Obviously,

$$\frac{\lambda_{min} + \lambda_{max}}{2} \leq \lambda_{max} \leq \lambda_{min} + \lambda_{max} \tag{5.6}$$

By (5.6) it is bounded from both sides by the proximity measure in (5.5) *and* has the eccentricity coded into it: When the value leans towards the left side of (5.6), eccentricity is low and vice-versa.

Thus, λmax is chosen as the raw saliency measure in our scheme.

This choice however, may still amplify locations which are very strong in terms of number of votes, but weak in eccentricity[2]. The product of E and λ_{max} produces the desired result, termed the *enhanced saliency* measure SM, or:

$$SM = \lambda_{max} \cdot (1 - \lambda_{min}/\lambda_{max}) = \lambda_{max} - \lambda_{min} \tag{5.7}$$

Thus, λmax-λmin is chosen as the enhanced saliency measure.

It is important to note that other functions of the eigenvalues can also satisfy the same conditions of monotonicity, but the ones chosen seem to be the *simplest* possible indicators of the desired behavior.

5.2.1.2 Detection of Junctions

A junction is defined as a *salient* point which also has *a low* eccentricity value.

Regular (non-junction) points along a curve are expected to have high eccentricity values. On the other hand, junction points are expected to have low eccentricity, since votes are accumulated from several different directions. By combining the eccentricity and the eigenvalue at a point, we acquire a continuous measure of the likelihood of that site being a junction. We redefine our previous definition of eccentricity slightly, so that low eccentricity scores high, or:

$$(E = \lambda_{min}/\lambda_{max}) \Rightarrow 0 \leq E \leq 1 \tag{5.8}$$

The product of our new eccentricity measure and the raw saliency measure λ_{max} yields the junction saliency operator:

$$E \cdot \lambda_{max} = (\lambda_{min}/\lambda_{max}) \cdot \lambda_{max} = \lambda_{min} \tag{5.9}$$

This process creates a *Junction Saliency map*. Interestingly enough, this map evaluates to just λmin at every site (as shown in Equation (5.9)), which simply means that the *largest non-eccentric* sites are good candidates for junctions. By finding all *local maxima* of the junction map we localize junctions.

---

2. For example, accumulation points and junctions! (where $\lambda_{min} \cong \lambda_{max}$)

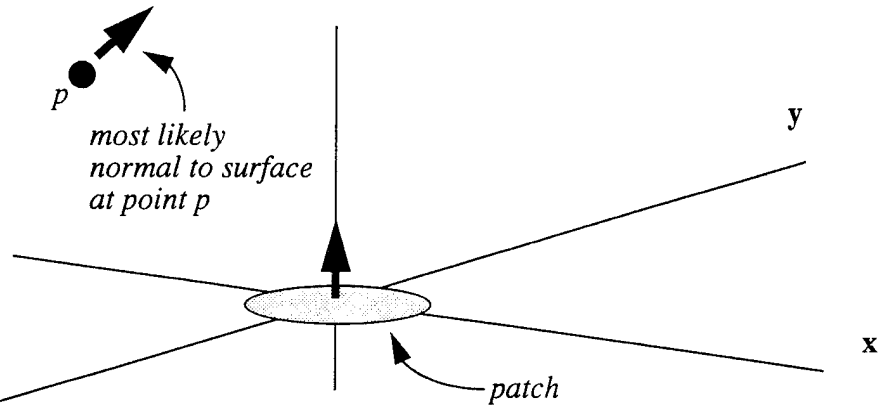


Figure 5.3 What is the most natural normal to a surface passing through point p and at the same time tangent to the patch at the origin?

5.2.2 3-D fields

In the 3-D case we would like to treat three elementary features, namely, a patch, a curve segment, and a point in space. A patch has a known 3-D normal, a curve segment has all possible normals lying on a plane, and a point in space has absolutely no directional data. We will construct a separate field for each of these features.

5.2.2.1 The construction of the Patch Extension Field

We assume that a patch with a known normal is available, and we ask the following question: for a given point in space, what is the most likely normal to a surface passing through that given point and also tangent to the original patch? Figure 5.3 illustrates that issue. It is clear that the desired normal at point p can be found by looking at a 2-D scenario, where both the origin and point p are on a plane. This reduces the problem to a 2D one, where the 2-D Extension field can be applied. Thus, constructing the 3-D Extension field is merely revolving the 2D Extension field around its vertical axis. This is illustrated in Figure 5.4 . Note that unlike the 2-D field, where each field element pointed *in* the direction of the most likely curve, in the 3-D case, each vector points in the direction of the *normal* at that location. This makes later stages of computation much simpler.

5.2.2.2 The curve segment Extension Field

Here we deal with a primitive with partial information regarding the orientation of a surface passing through it. All we know is that the given segment has to lie on the desired surface. Again we ask the question: What is the most likely surface to pass through a point p in space and have the segment at the origin lying on it? The answer is very simple. A segment and a point in space define exactly one plane[3]. And since a

3. Except, of course, the points co-linear with the segment.

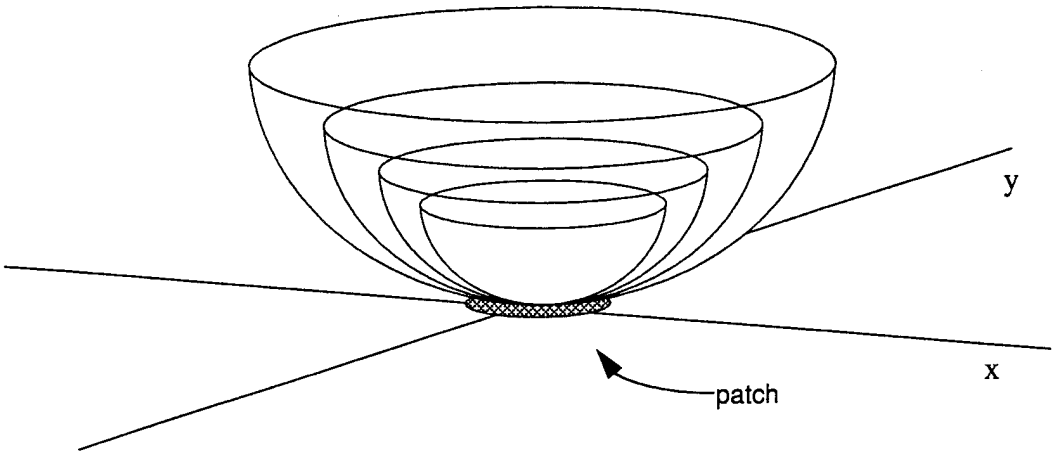


Figure 5.4 The general shape of the 3-D Extension Field. The lower part is omitted from the sketch, but is similar to the upper part. Field elements are normal to the surfaces shown, and were also removed for display purposes.

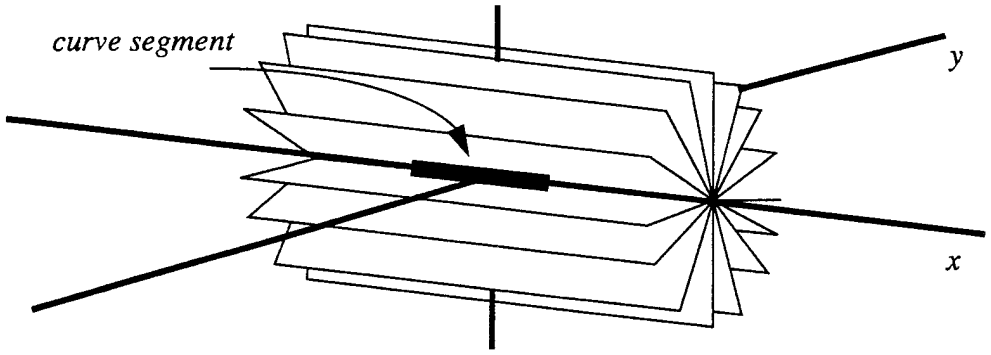


Figure 5.5 The general shape of the curve segment field. All planes go to infinity, with diminishing strength. The field elements are in reality normals to the drawn planes

plane is the best surface in terms of the perceptual constraints, it is also the most likely to appear.

A practical way of constructing this field is to take the Patch Extension field and convolve it with a multi-directional patch[4]. This last operation is similar to revolving the patch Extension field around itself along the x (or y) axis (referring to notation in Figure 5.4). By symmetry considerations, it is simple to show that the resulting field will have the correct orientations everywhere in space (as shown in Figure 5.5). This construction also determines the strength values at every site of the field.

5.2.2.3 The 3-D Point field

The 3-D point field is even simpler to derive. The only thing in common to all surfaces passing through a point in space and the origin, is that the line connecting the

4. The 2-D point field was constructed by the exact same way from the 2-D Extension field (in [63]).

two points is on all of them. However, in this case, there does not exist a single maximum likelihood normal. That is, at each point in space, *many* normals are equally likely. Luckily, they all lye on a plane perpendicular to the line formed by the point in question and the origin. We thus choose to describe the contribution of all these normals with a single 3-D vector pointing in the direction of the above line. We will later show how such a voting vector is treated to determine saliencies.

5.2.3 Directional Convolution

The process of computing the saliency maps is similar to the 2-D case. We will describe it here again for sake of completeness. Computing the Saliency maps can be thought of as a directional convolution with one of the above fields (mask). The resulting map is then a function of a collection of fields, each oriented along a corresponding short normal in 3-D. The whole operation is performed in a 3-D grid or array. Each site accumulates the 'votes' for its own preferred orientation and strength from every other site in the image. These values are combined at a site as described next.

When the input data consists of non-oriented features (e.g. 3-D points), a 2-pass convolution was found to work best, first applying the point field in order to estimate orientations, and then the patch Extension field, for the final results. The same procedure is performed if curve primitives are present in the input image.

5.2.4 Combination at each Site

Combination per site is really the process of choosing, for each site, the preferred normal that will show up in the final saliency map. The 3-D case will be derived based on the same methodology used for the 2-D scheme (as described earlier in this paper).

5.2.4.1 The 3-D case

Here we need to consider a 3x3 variance-covariance matrix, as shown in Equation (5.10), where λmax, λmid, and λmin signify the three sorted eigenvalues of the system. (Note that the 3-D discussion assumes that normals to the desired surfaces are doing the actual voting[5]!)

$$\begin{bmatrix} m_{200} & m_{110} & m_{101} \\ m_{110} & m_{020} & m_{011} \\ m_{101} & m_{011} & m_{002} \end{bmatrix} = \begin{bmatrix} EVmin \\ EVmid \\ EVmax \end{bmatrix} \begin{bmatrix} \lambda min & 0 & 0 \\ 0 & \lambda mid & 0 \\ 0 & 0 & \lambda max \end{bmatrix} \begin{bmatrix} EVmin^T & EVmid^T & EVmax^T \end{bmatrix} \quad (5.10)$$

The three eigenvectors will correspond to the three principal directions of an ellipsoid in 3-D, while the eigenvalues describe the strengths and agreement measures of the 3-D votes.

---

5. Except for the non-oriented case, which is discussed in a separate section.

As before λmax is bounded on both sides by the sum of eigenvalues (which corresponds to raw strength) and at the same time encodes the eccentricity. When it leans toward the right hand side of Equation (5.11), eccentricity is high and when it leans toward the left hand side, eccentricity is low.

$$\frac{\lambda_{min} + \lambda_{mid} + \lambda_{max}}{3} \leq \lambda_{max} \leq \lambda_{min} + \lambda_{mid} + \lambda_{max} \tag{5.11}$$

Thus, λmax is selected as a raw saliency measure for surface normals, and the corresponding eigenvector determines the orientation of that normal.

To further enhance the measure we can require that the other two eigenvalues be low compared to the λmax. This can be achieved by looking at the difference, λmax - λmid. The expression will yield high values only when both λmid and λmin are small. The most likely normal to the surface, is merely the eigenvector corresponding to λmax.

The same logic holds for intersections between surfaces. Here, we would like to look at λmid as a saliency measure. When it is high, so must λmax, and the location is really characterized by votes coming from exactly two separate surfaces.

Thus, λmid is chosen as a raw saliency measure for intersection between surfaces.

Again, this measure can be enhanced by considering λmid - λmin. This last expression will exclude locations along intersection curves that belong to a higher-level intersection(i.e. a junction). The direction of the curve is given by the eigenvector perpendicular to the two surfaces, or the one corresponding to λmin.

Lastly, we claim that large values of λmin will correspond to locations where three (or more) smooth surfaces intersect, or a 3-D junction. It is clear that if λmin is large, so are the other two. Three large eigenvalues describe a spherical distribution of normals, meaning normals from many different orientations have voted for that point in space.

λmin is thus chosen as the junction saliency map.

The 2-D and the 3-D results can be summarized in Table 6:. The highlighted col-

Table 6: 2-D and 3-D results

Feature	2-D raw saliency	2-D enhanced saliency	3-D raw saliency	3-D enhanced saliency
Junction	λmin	λmin	λmin	λmin
curve	λmax	λmax - λmin	λmid	λmid - λmin
surface			λmax	λmax - λmid

umns emphasize a somewhat surprising correspondence between the cardinality of the feature and the eigenvalues.

As usual, a disclaimer is in order. The above heuristic approach is by no means the only (or the best) indicator of saliency. We believe that it is one of the simplest to implement, is fairly intuitive, and proves to behave well as an indictor of saliency.

5.2.4.2 Combination per site for the Point field - 3-D case

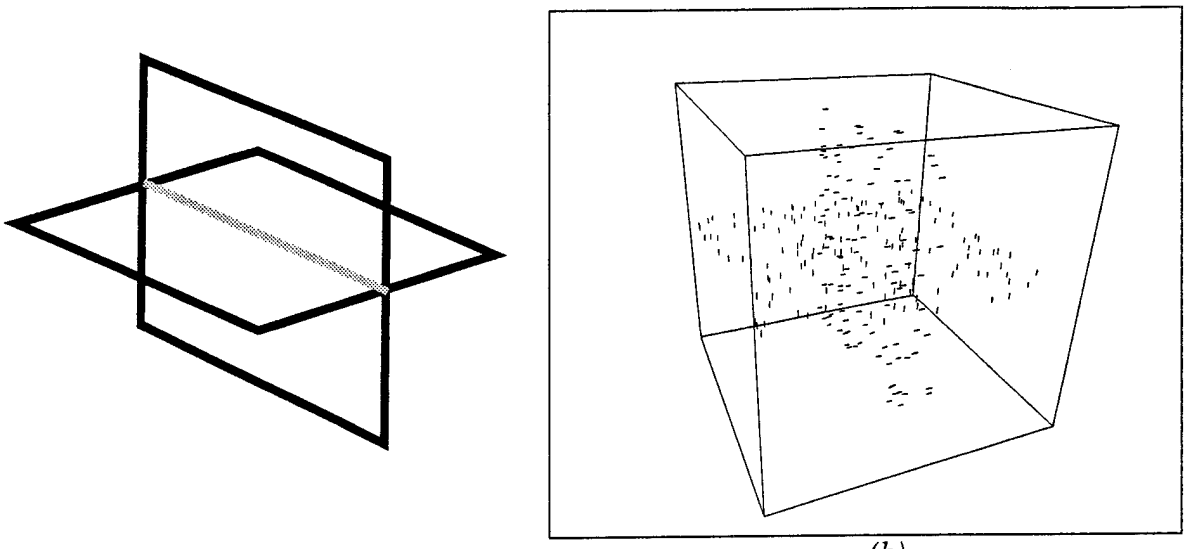
when only 3-D points are available (no orientation), we first attempt to find a maximum likelihood normal to those points, with the aid of the Point field. Again we compute the 3 eigenvalues and eigenvectors, but a different interpretation is now needed. Recall that we selected a vector lying along the two points to represent all possible surfaces.

Recovering surface normals requires merely to select the eigenvector corresponding to the smallest eigenvalue. This orientation will be the vector perpendicular to the best plane described by the ellipsoid of votes.

In order for a certain location to be a good candidate for a surface, the votes have to distributed in such a way that they create a "flat" sphere. This can obviously be tested by looking at λmid - λmin. Large values of that term will indicate high likelihood of a surface passing through the location.

The above procedure assigns orientations to the given set of input points. It could also assign strength to points, thus reducing the influence of noise *before* the second pass.

Unfortunately, it is impossible at this stage to recover intersections and junctions from the computed maps. It is necessary to perform a second convolution using the Patch Extension field, on the input data, which now has orientation data available. Obviously, the final results when a cloud of non-oriented points is given, are not as good as with oriented input data.



(a) (b)

Figure 5.6 Input consists of randomly selected set of points lying on the two intersecting surfaces. (a) A schematic model of input (The lighter line denotes the intersection between surfaces). (b) Projection of input samples.

5.2.5 Noise tolerance

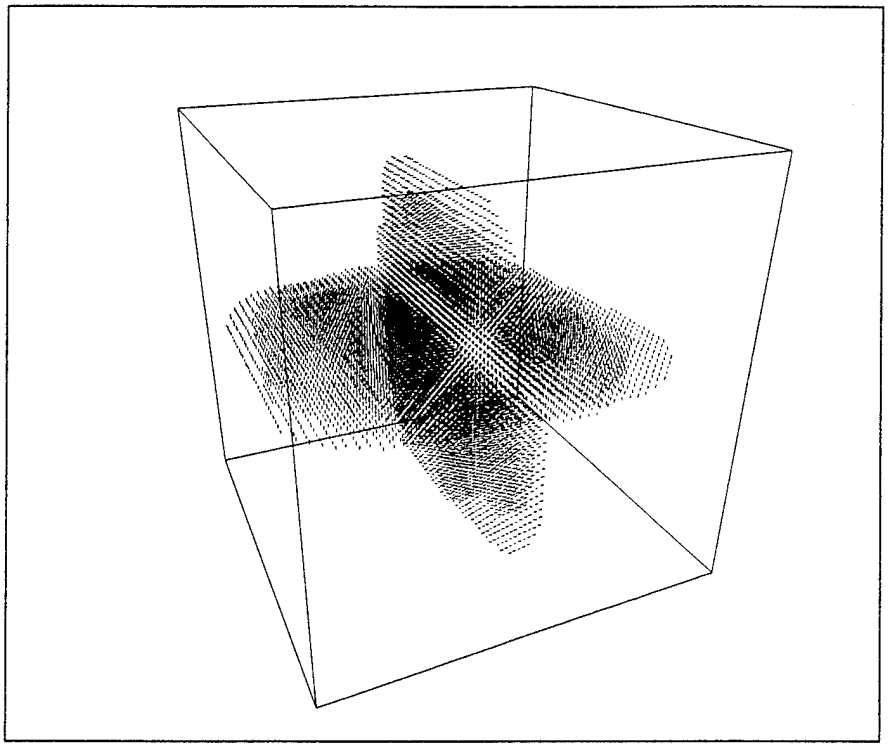
Similar to the 2-D case, the scheme is not sensitive to noise in the form of erroneous features, or localization errors of the measurements, since a voting scheme is employed. Also, a priori distribution of noise is expected to be directionally uniform, such that computed orientations are not corrupted. Saliency selectivity[6], however, could suffer when noise is present.

5.3 Results

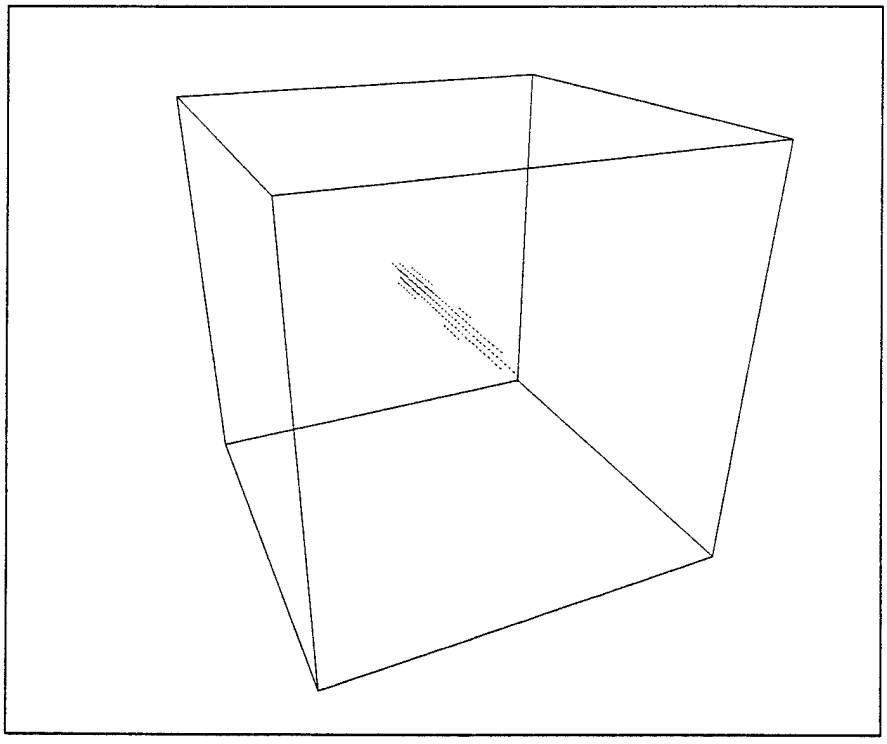
We have generated some synthetic images to test our scheme. The first example consists of two planes positioned in space as shown in Figure 5.6 . We randomly sampled the planes of Figure 5.6 . Grid size was 50X50X50, each plane has ~100 samples. Since 3D saliency maps are 4D in nature[7], we thresholded (for display purposes) all maps to a point where the 2D projection becomes legible, and small line segments denote the orientation at each site In practice one would like to follow the dense saliency maps, and extract a description of each surface[8]. The task of following the surfaces along the saliency maps was not preformed in this work. Figure 5.7 shows the saliency maps for surfaces and curves. Note that the maps are dense now, and every site contains a normal. Also, the orientation of the segments in the intersection map (Figure 5.7 (b)) are pointing in the direction of curve.

---

6. i.e., the variance between figure and ground in the saliency maps.
7. Strength and orientation at each 3D site.
8. e.g. by a triangulation.

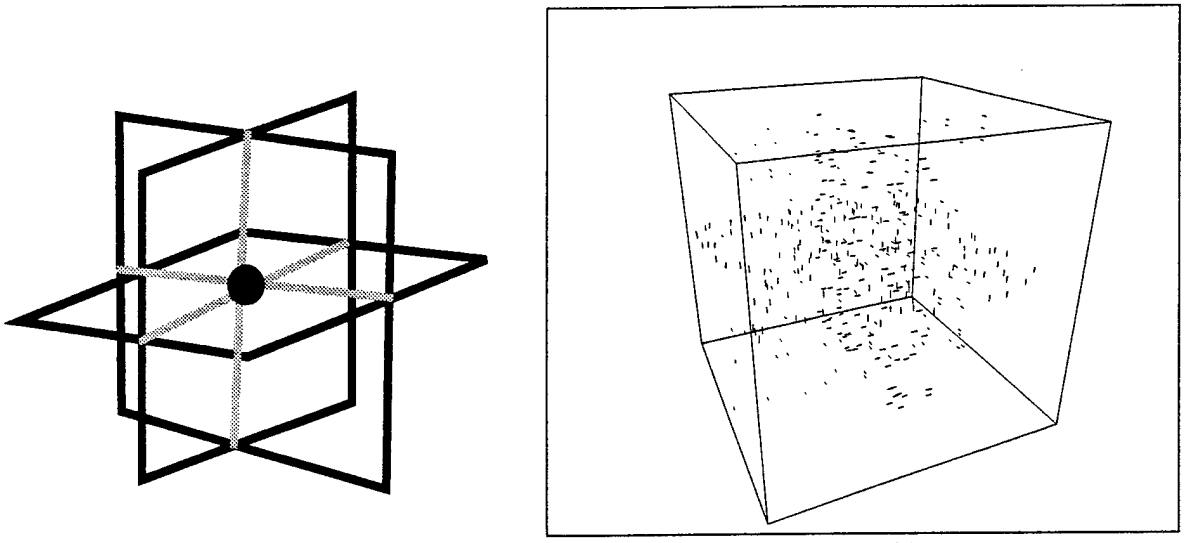


(a)



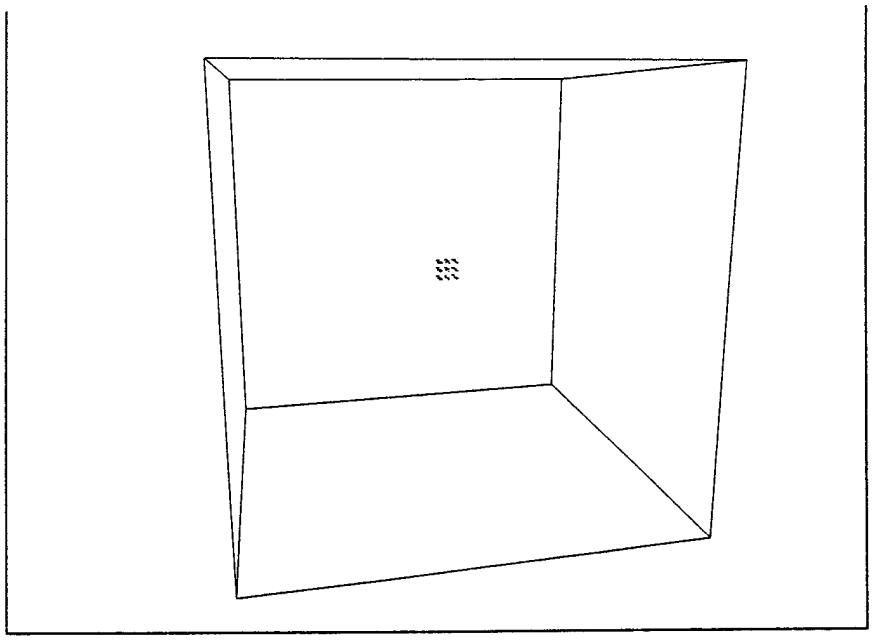
(b)

Figure 5.7 (a) surface saliency map. (b) Curve saliency map.



(a) (b)

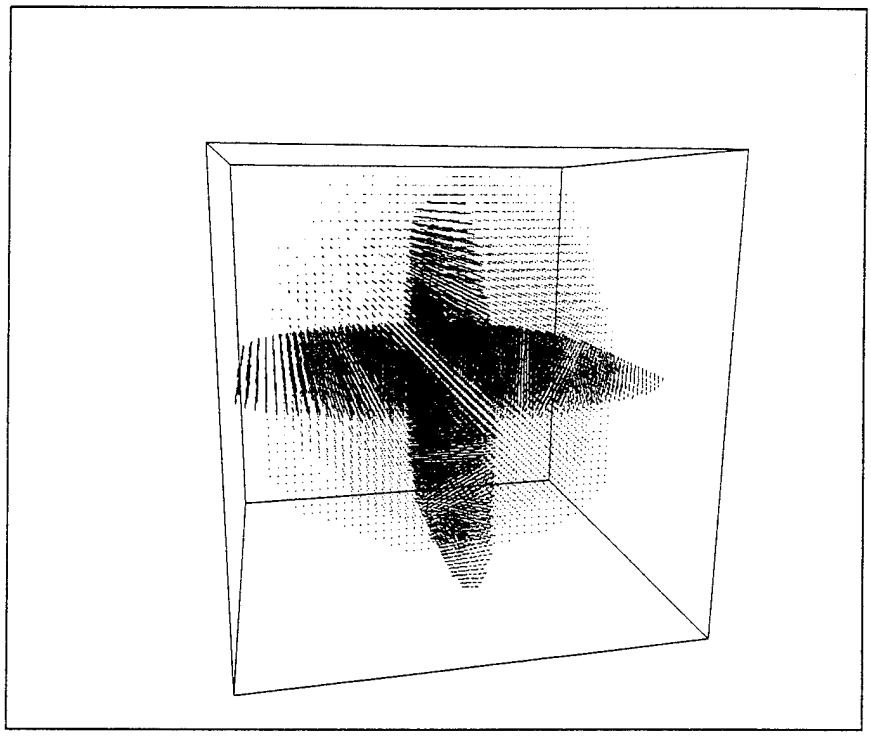
Figure 5.8 (a) A schematic model of input (The brighter lines denote the intersections between surfaces, and the dot is the 3D junction). (b) Projection of



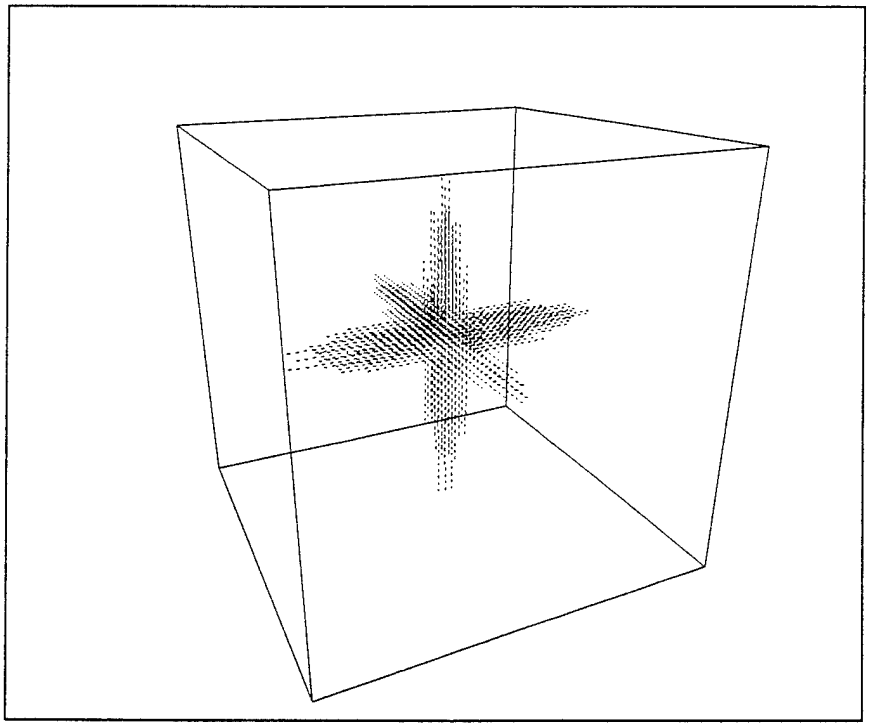
(c)

Figure 5.9 (continued)(c) Junction saliency map.

Figure 5.6 describes a scenario with three intersecting planes. Figure 5.9 shows the three corresponding saliency maps. Both the surface map and the intersection map tend to decay toward the edges of our 3D space. This is due to the limitations

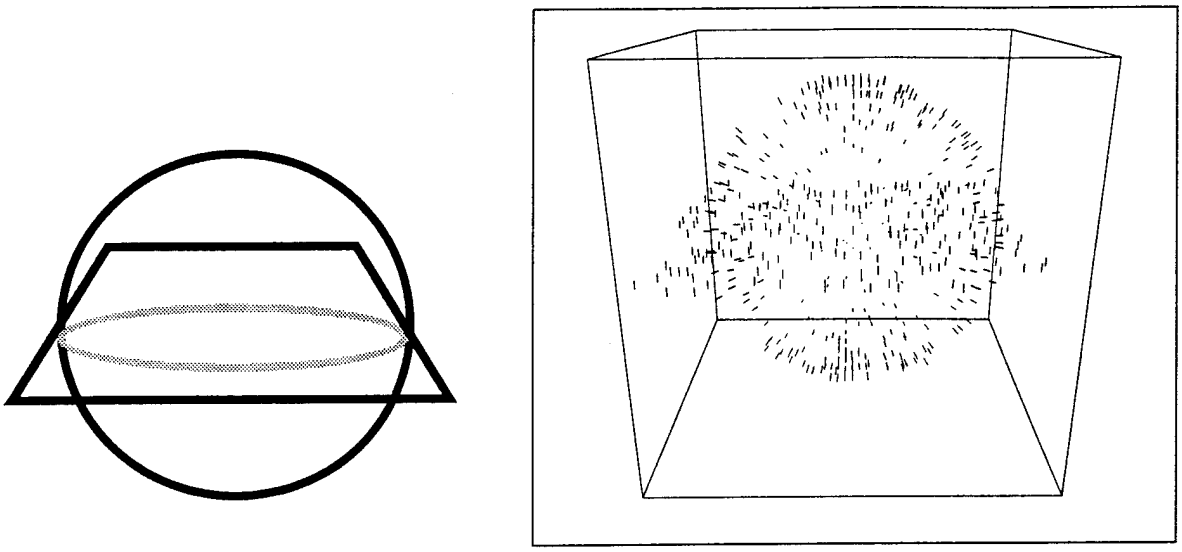


(a)



(b)

Figure 5.9 (a) surface saliency map. (b) Intersection saliency map.



(a) (b)

Figure 5.10 (a) A schematic model of input (The brighter lines denote the intersections between surfaces). (b) Projection of input samples.

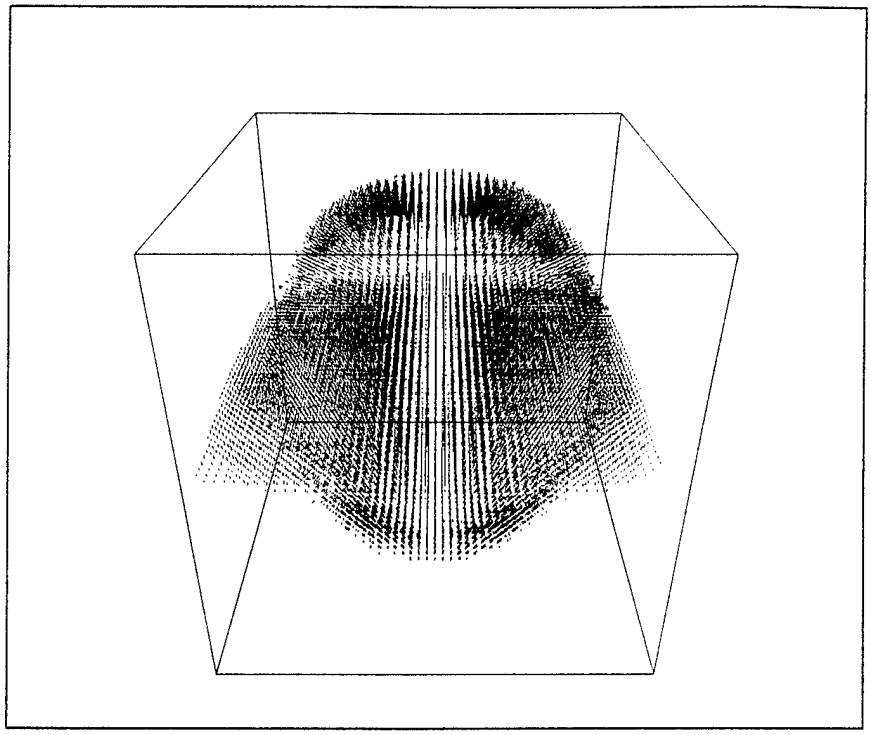
of our display and the application of a constant threshold to the data. Figure 5.10 depicts an example with curved surfaces. Here a sphere is intersected with a plane. As before, the plane consists of ~100 measurements, and the sphere has about 200 measurement points. Results are shown in Figure 5.9 .

5.3.1 Noise tolerance

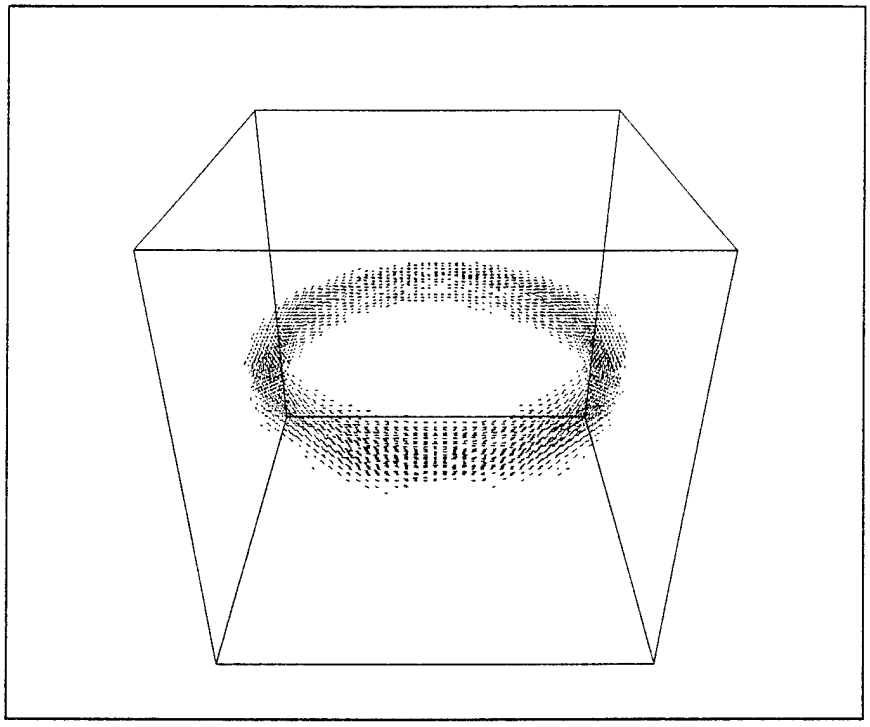
We choose a simple curved surface to illustrate the noise immunity of the scheme. A part of a sphere is chosen, and ~150 points are randomly selected on the sphere, as shown in Figure 5.12 . We 'sprinkle' the space with an increasing number of erroneous segments. The results in Figure 5.7 show the surface saliency maps with 125,250, and 375 additional random segments. It is easy to see that they virtually the same. The input set with 250 noisy points is shown in Figure 5.12 (b) for reference.

Finally, we show an example where the input consists of a cloud of non-oriented points, with a considerable amount of noise. Again a quarter of sphere is embedded in noise. The sphere has ~200 data points, and ~100 noise points, as shown in Figure 5.14 (a).

The first phase is to compute normals to the existing input points. This is done by convolving with the 3D Point field. Figure 5.14 (b) shows the result of the first phase. note that not only do the points have orientation vectors attached to them, but many of the noise points have been attenuated. The second phase is the standard Patch extension field convolution. The final result is shown in Figure 5.14 (c)l.

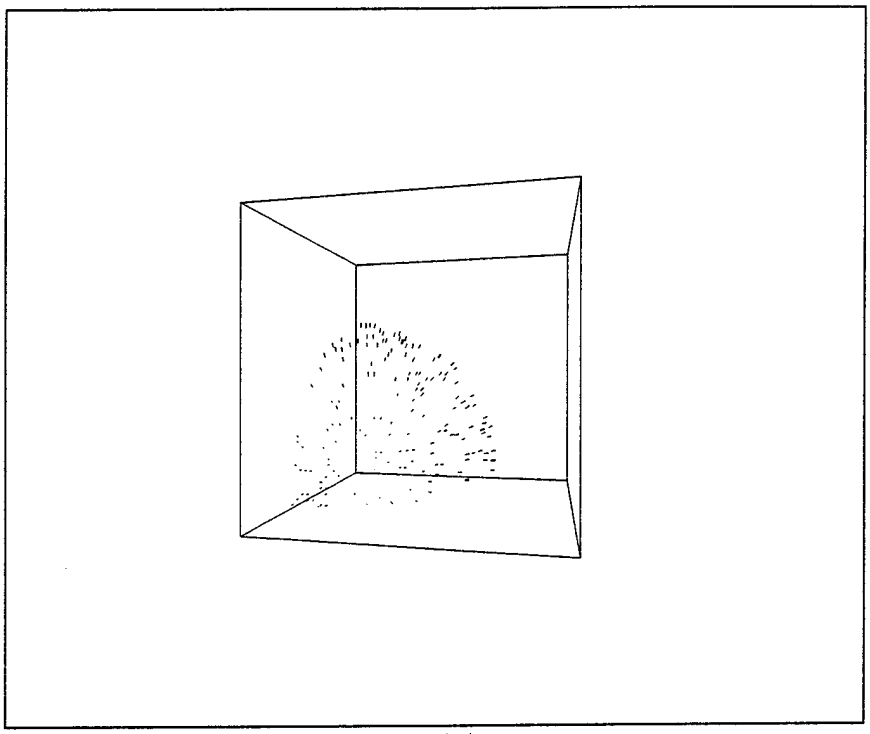


(a)

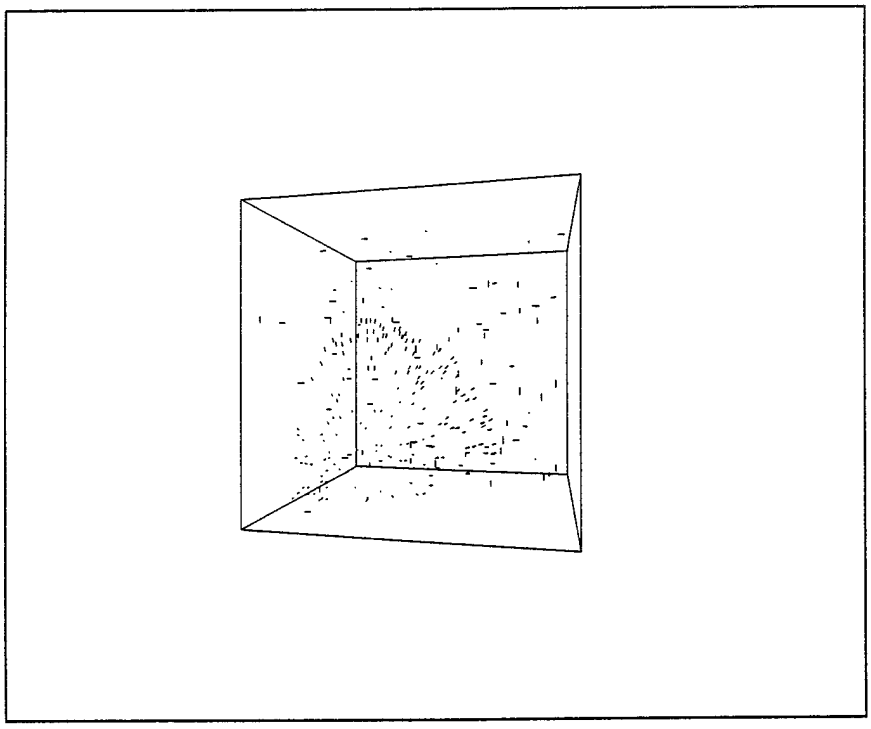


(b)

Figure 5.11 (a) surface saliency map. (b) Intersection saliency map.

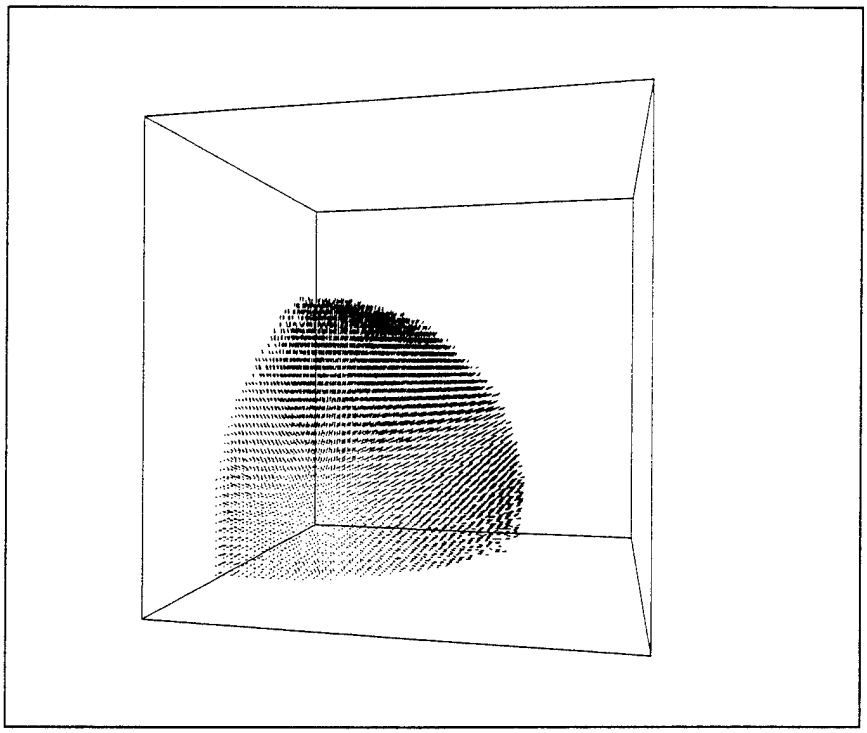


(a)

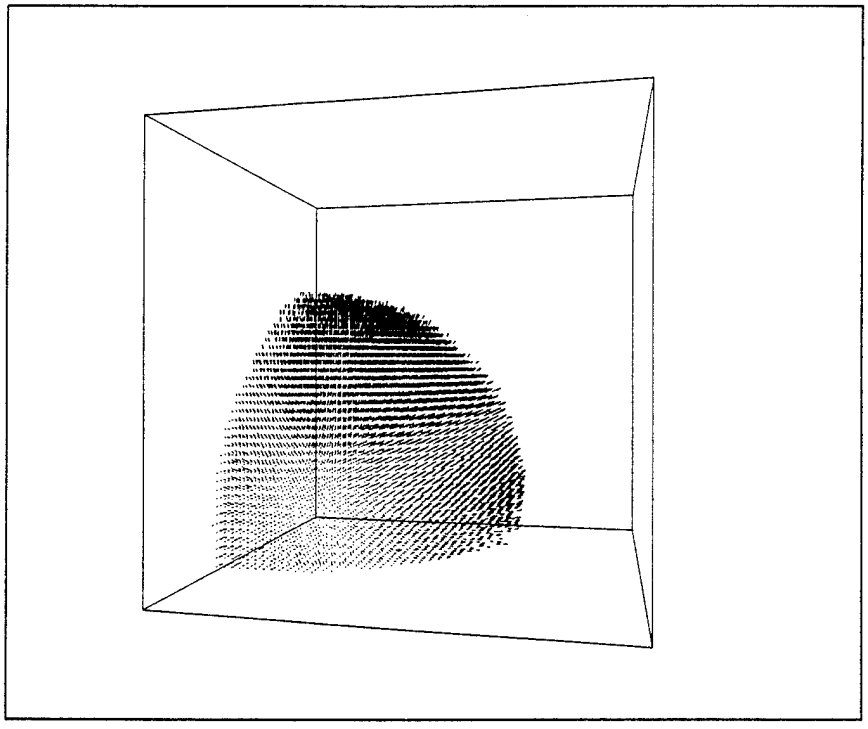


(b)

Figure 5.12 (a) Sample points of a quarter sphere centered at the origin.(b) Same sphere embedded in 250 noisy points.

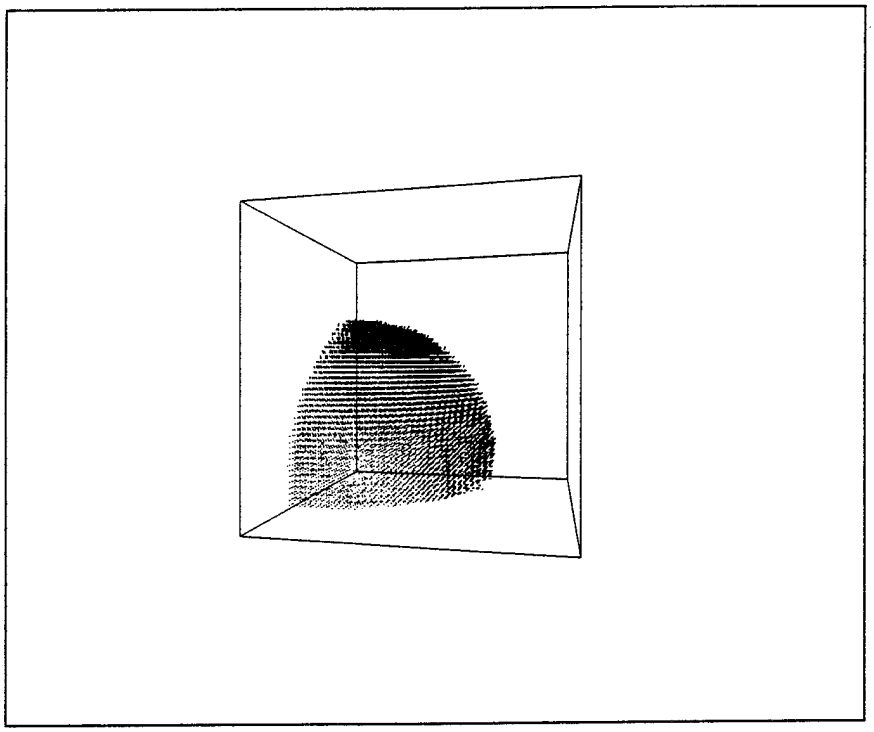


(a)

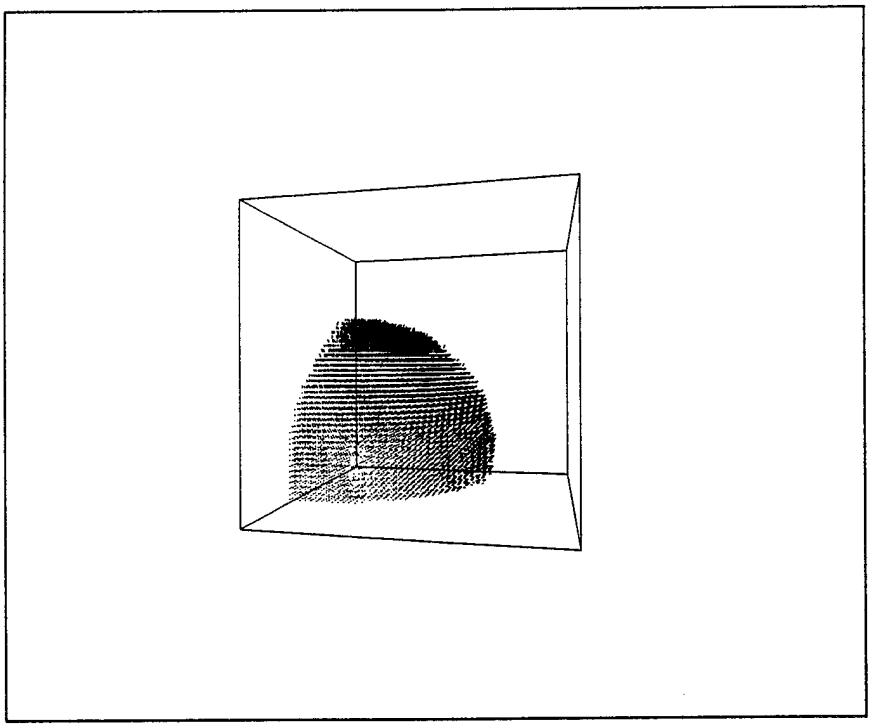


(b)

Figure 5.13 (a) surface saliency map without noise. (b) with 125 noise segments.

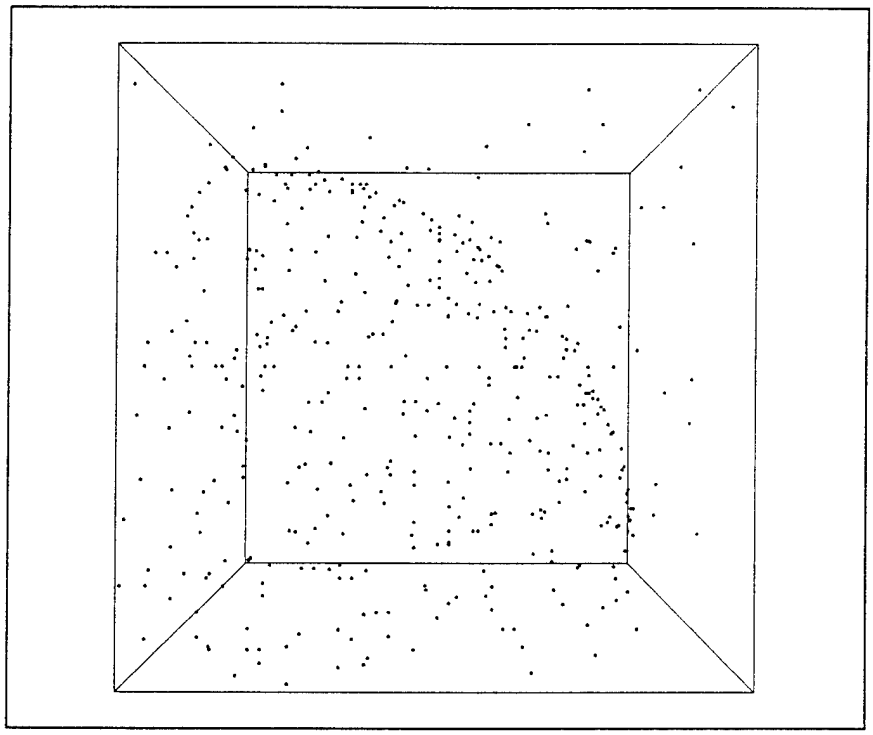


(c)

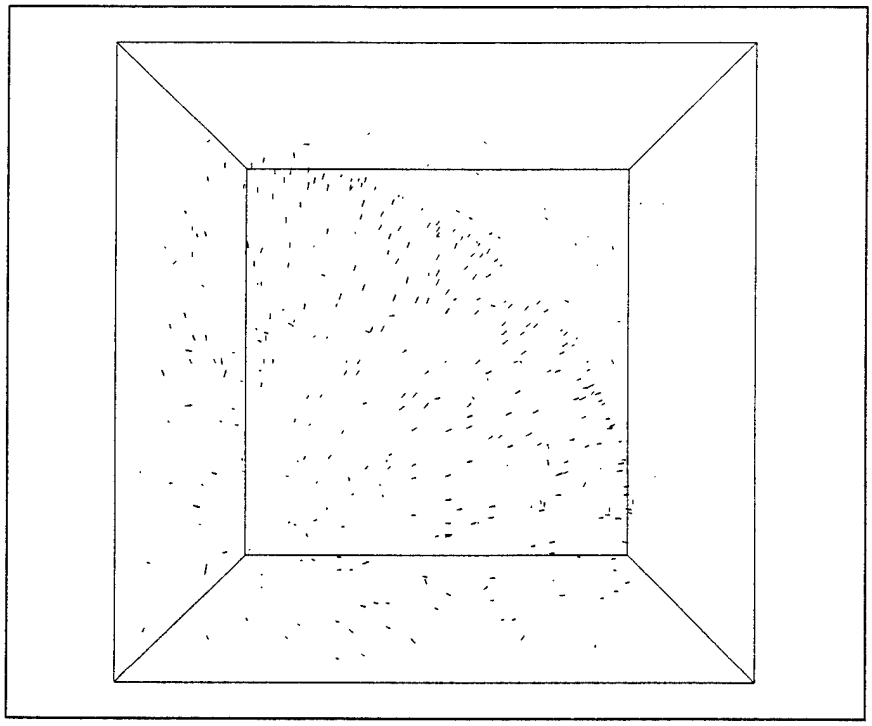


(d)

Figure 5.13 (continued) (c) with 250 segments. (d) with 375 noise segments.

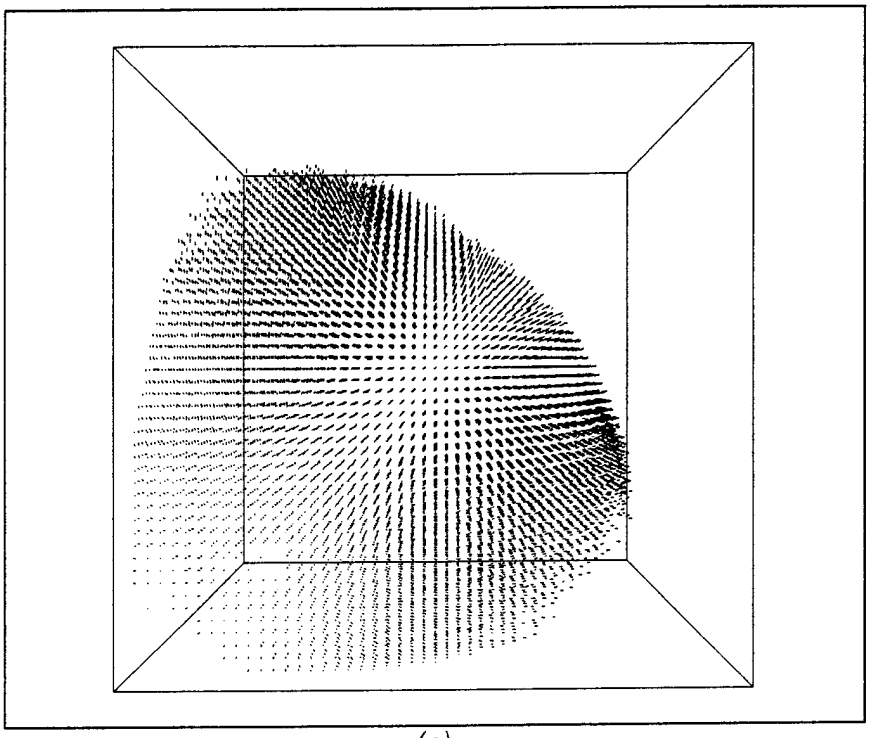


(a) Input Image



(b) With estimated normals and strength

Figure 5.14 Results of saliency analysis



(c)

Figure 5.14 (continued) (c) Final saliency map

5.4 Conclusion

We have presented a method to recover surfaces, intersection between surfaces, and 3-D junctions by applying perceptual grouping rules. The method presented is an extension of a 2-D approach proposed earlier by the authors, and uses a non-iterative and parameter-free algorithm. The method can handle scenes with any number of objects, each having an arbitrary genus number, without any a priori knowledge. In particular, an initial guess is not needed.

The complexity is $O(n^3k)$ in general, where n is the side size of the volume, and k is the number of available measurements. Some practical short-cuts can reduce the complexity further. The algorithm is highly parallel in nature, and as such can be easily implemented on a parallel machine.

5.5 References

[59] N. Ahuja and M. Tuceryan, *Extraction of early perceptual structure in dot patterns: integrating region, boundary, and component Gestalt*, CVGIP 48, 1989, pp. 304-356.

[60] J. Dolan and R. Weiss, *Perceptual Grouping of Curved Lines*, Proc. IUW89, Palo Alto, CA., pp. 1135-1145.

[61] P. Fua and P. Sander, Segmenting Unstructured 3D Points into surfaces, ECCV 92, Santa Margherita Ligure, Italy, May 1992, pp. 676- 680.

[62] G. Guy and G. Medioni, *Perceptual Grouping using Global Saliency enhancing operators*, Proc. of ICPR92, The Hague, Holland, 1992, pp. 99-104

[63] G. Guy and G. Medioni, *Perceptual Grouping using Global Saliency enhancing operators*, IRIS-USC Technical report, *to appear*

[64] G. Guy and G. Medioni, *Inferring Global Perceptual contours from Local Features*, Proc. of CVPR93, New-York, New-York, 1993, pp. 786-787.

[65] M. Kass, A. Witkin, and D. Terzopoulos, *Snakes: Active Contour Models*, in *International Journal of Computer Vision,* January 1988, pp.321-331.

[66] C. Liao and G. Medioni, *Surface Approximation of a Cloud of 3-D Points,* CAD94 workshop, Pittsburgh, PA.

[67] D.G. Lowe, *Three-dimensional object recognition from single two-dimensional images*, Artificial Intelligence 31, 1987, 355-395.

[68] R. Mohan and R. Nevatia, *Segmentation and description based on perceptual organization*, Proc. CVPR, Jun. 1989, San Diego, Ca., pp. 333-341.

[69] R. Mohan and R. Nevatia, *Using Perceptual Organization to Extract 3-D Structures*, IEEE Trans. on PAMI, Vol. 11, No. 11, November 1989, pp. 1121-1139.

[70] D. Terzopoulos, A. Witkin, and M. Kass, *Constraints on deformable models: Recovering 3D Shape and Nonrigid Motion, Artificial Intelligence,* Vol. 36, 1988, pp. 91-123.

6 From an Intensity Image to 3-D Segmented Descriptions

Mourad Zerroug and Ramakant Nevatia

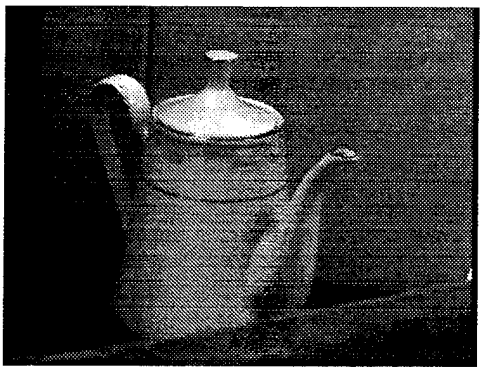
We address the inference of 3-D segmented descriptions of complex objects from a single intensity image. Our approach is based on the analysis of the projective properties of a small number of generalized cylinder primitives and their relationships in the image which make up common man-made objects. Past work on this problem has either assumed perfect contours as input or used 2-dimensional shape primitives without relating them to 3-D shape. The method we present explicitly uses the 3-dimensionality of the desired descriptions and directly addresses the segmentation problem in the presence of contour breaks, markings shadows and occlusion. This work has many significant applications including recognition of complex curved objects from a single real intensity image. We demonstrate our method on real images.

6.1 Introduction

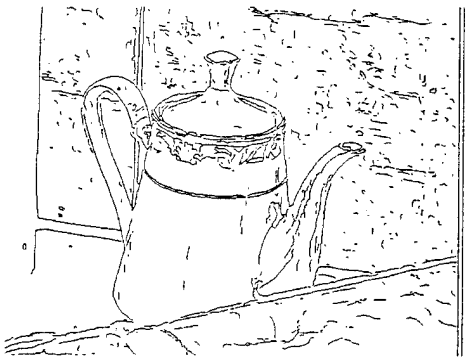
Recovering and representing shape of a complex object is one of the most fundamental tasks in computer vision. A good shape representation is useful not only for recognizing an object but also in determining how to manipulate it, how to navigate around it and to learn about new objects.

We believe that a good way to represent a complex object is by decomposing it into parts and describing the parts and the relationships between them. If the parts are complex, they can be decomposed into simpler parts and described in the same way as the larger object. Further, we believe that the parts should be described as volumetric primitives. Such a representation is very rich, stable and allows us to handle occlusion and articulation in a natural way.

Use of simpler parts to describe more complex objects has a long history in computer vision [75,83,92]. Biederman has argued that a similar scheme is used by the human visual system as well [74]. However, in spite of these theories and the obvious advantages of segmented (or part/whole) representations, their use in computer vision systems has been limited. We believe that this is due to the difficulty of actually computing segmented shape description from real data. The part decomposition hierarchy is not given in advance, we must infer it from the observable features in the data. Most of the previous work has used range data. In early work [87], Nevatia and Binford used perfect contours derived from range data. Pentland [88] used range data to segment objects into super-quadric primitives. Surface based segmentation using range data has been studied by several researchers [73,79].



(a) intensity image



(b) edge image

Figure 6.1 Sample real image of a compound object. .

In this paper, we focus on computing segmented volumetric descriptions from a single intensity image. This is a task that humans perform effortlessly. It is also important for computer vision as a single image can be acquired rather easily, without extensive control of illumination or elaborate calibration procedures. Using single intensity images does pose many problems, however. Lack of direct 3-D measurements makes it more difficult to determine discontinuities that may characterize part boundaries. Instead, we must work with intensity boundaries which may correspond to depth boundaries, but also to markings, shadows, specularities and noise. Furthermore, object boundaries are unlikely to be complete due to both poor edge localization and occlusion. These characteristics make the techniques developed for range data and perfect contours [72,77] largely unapplicable to the case of intensity images. Detection of concavities by computing curvature extrema, for example, as used in [72] is not possible; it is most likely that such extrema are missing in lines extracted from an intensity image.

Figure 6.1 shows an example. Notice that the boundaries are not all perfect, continuous or even part of the outline of the object (most are not in fact). Also, notice that the pot is partially occluded by both the spout and the flat object in front which also occludes part of the spout. The pot itself partially occludes the handle whose ends are not visible. Here, we would like to separate the teapot from the background and describe it as consisting of the arrangement of four parts: the conical pot, the lid, the spout and the handle. Deciding that there is such an object and with that composition is a non-trivial problem. Moreover, we would like to recover the 3-D shape of the object.

Some previous work has attempted to address part and object segmentation from intensity images. Specifically, the work of Rao and Nevatia [90], and Mohan and Nevatia [84] has attempted to solve similar problems to those presented in this paper. However, these efforts relied largely on heuristic properties of observed contours and did not attempt any 3-D recovery (they address a 2-dimensional problem). The method of [81], based on a neural network implementation, addresses geon-based descrip-

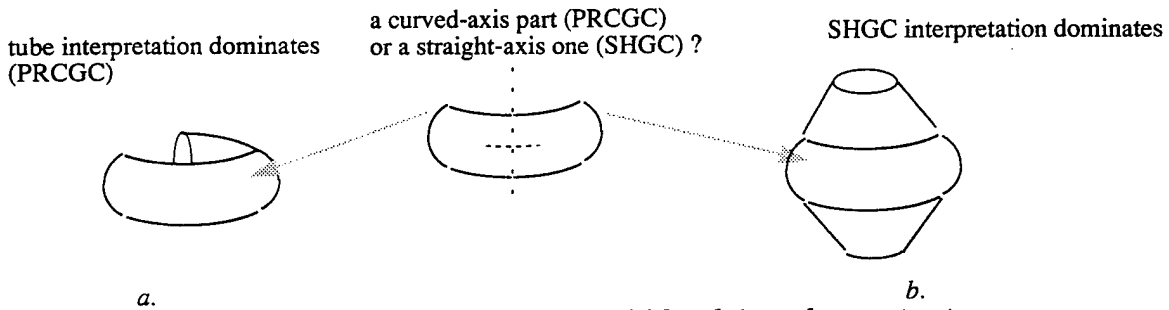


Figure 6.2 Part perception could be driven by context

tion and recognition from possibly discontinuous boundaries. The boundaries were synthetic and the axial descriptions were assumed given. In this paper, we present an approach that is more closely connected to rigorous properties of contours of 3-D objects to solve the figure/ground and part segmentation problems, *and* to recover 3-D structure of the objects.

In this work, we have chosen generalized cylinders as primitives for part description. The classes of GCs we allow here are the *straight homogeneous generalized cylinders* (SHGCs) and *planar right generalized cylinders* (PRGCs). SHGCs are obtained by scaling a planar cross-section along a straight axis curve. PRGCs are obtained by scaling a planar cross-section along a curved planar axis curve. More precisely, two sub-classes of PRGCs are addressed: *planar right constant generalized cylinders* (PRCGCs), characterized by a constant sweep, and circular PRGCs, characterized by a circular but varying size cross-section. We believe that a combination of these classes of GCs can represent well a large fraction of man-made objects.

Our approach to detecting and describing complex objects is based on the exploitation of the projective properties of the above classes of parts and of their relationships. They consist of geometric *invariant* and *quasi-invariant* and structural properties of the image boundaries of an object (the projection geometry is approximated by orthography in this work). We have used a similar approach earlier to analyze scenes of objects consisting of *single* GC primitives. Our work on SHGCs is described in [97,98,99]. Our analysis of circular PRGCs is presented in [96,97]; the method for recovering them from a single intensity image is submitted separately to this conference. Dealing with complex objects introduces many new difficulties due to the interactions between parts such as non-visibility of cross-sections, or substantial (self) occlusion, and ambiguities inherent to compound objects. For example, a part's perception depends not only on the properties of its boundaries but on the surrounding structure as well (which could be thought of as its context [71]). An example is given in Figure 6.2 . The middle part (the same set of boundaries) has different interpretations in the right and left drawings.

Our method for object segmentation and description consists of two main levels, the *part level* and the *object level*. In this approach, the figure-ground discrimination

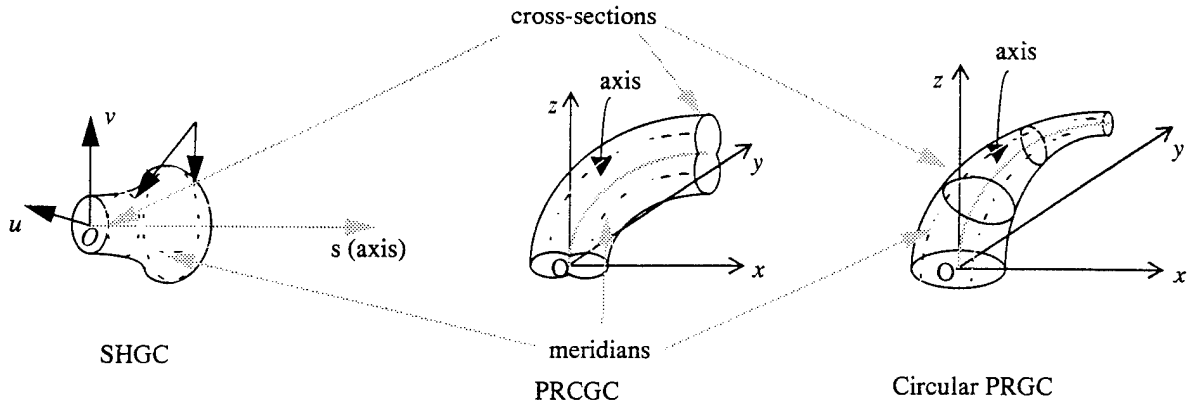


Figure 6.3 Generalized cylinders used as parts in our approach

and shape description are cooperative rather than sequential processes. The projective properties we use also help us recover the 3-D shapes of the parts we detect.

In this paper, we apply our method to a restricted (but common) class of compound objects, namely those which consist of two possible types of joints between parts: end-to-end and end-to-body. In the former, one part's end is in contact with the other part's end and in the latter, one part's end is in contact with the other part's body. Our method and some results are described in the following. First, we provide a brief overview of our part detection and description system and then describe our method for inferring the joints and the compound object.

6.2 The Part Level

Most of the methods used in this level have been described elsewhere [96,97,98,99]; due to lack of space, we will summarize them instead of giving details. The classes of parts addressed in this work (SHGCs and PRGCs) are shown in figure Figure 6.3 .

Two fundamental aspects characterize our method for detecting parts. First, it uses geometric projective (orthographic) invariant and quasi-invariant, and structural, properties of the above classes of GCs. Second it organizes the segmentation and description as a hypothesize-verify process. The projective properties provide *necessary* conditions that projections of SHGCs and PRGCs must satisfy in the image. They also give direct relationships between 3-D shapes and computable image descriptions which is useful for recovering volumetric descriptions from a monocular image. Finally, in using view invariant (and quasi-invariant) properties, the method and the descriptions it produces do not depend on the particular viewpoint the scene is viewed from.

The method for detecting parts is structured in three sub-levels: the curve level and symmetry level and the surface patch level (see figure Figure 6.4). The curve level consists of forming boundaries from image edges. The next level is concerned with

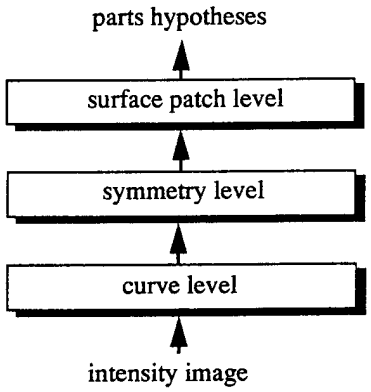
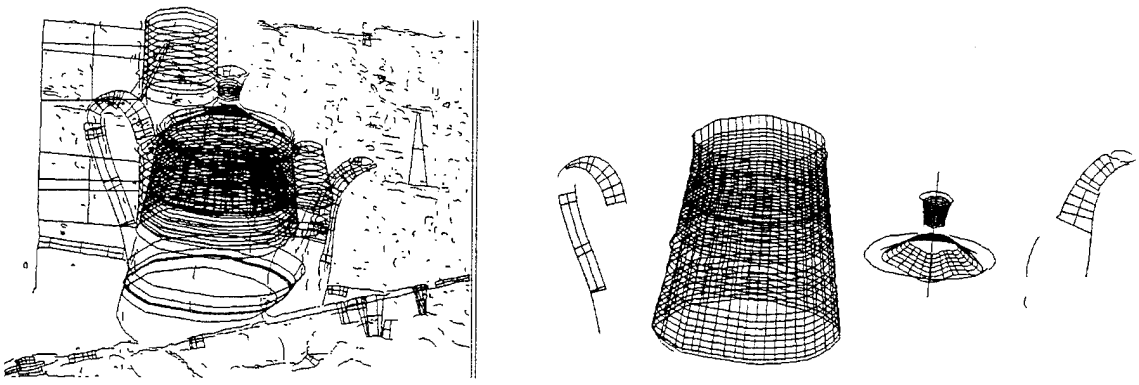


Figure 6.4 Black diagram of the part level

forming parallel symmetry [93] relationships between the boundaries (parallel symmetries are one of the invariant properties of the primitive parts). The symmetries are used to initiate the search for parts. The surface patch level is intended to form part descriptions using the boundaries and symmetries formed in the previous levels. It consists of a hypothesize-verify process of several steps: *detection* of local surface patches, *grouping* of local surface patches and *verification* of parts hypotheses. In the detection step, the projective properties are locally applied between pairs of image boundaries. Groups of boundaries which (locally) verify the properties are hypothesized to correspond to portions of parts. In the grouping step, local surface patches which are likely to project from the same scene object are merged to form parts hypotheses. The grouping criteria are based on the similarity of their projective descriptions. For example, for an SHGC, the local surface patches must have the same axis projection. The verification step consists of a filter which rules out inconsistent parts hypotheses. Consistency is defined in terms of both geometric and structural criteria. The geometric criteria consist of enforcing global consistency of the geometry of the part with respect to its geometric projective invariants and quasi-invariants. The structural criteria consist of enforcing closure and associated junctions at the end of a part. They express the fact that the image of a part may have one of several well defined closure patterns involving specific junction labeling (that include occlusion junctions) [82].

This hypothesize-verify nature of the part detection method allows us to handle markings, shadows and occlusion. Non-object boundaries (such as surface markings and shadows) are unlikely to survive the successive application of the strong projective properties. But some regular markings might still survive the verification tests.

Several enhancements, beyond our previously described work, have been made to the part level in order to handle compound objects. First, cross-sections may not be visible due to joints between parts. Second, by using a more complete set of projective properties (those of joints are described in section 6.3.1), several ambiguities occur and need to be addressed. The ambiguities are due to the fact that different 3-D



a. initial local surface patch hypotheses (146 hypotheses; only a few are shown here so as not to clutter the image)

b. verified parts hypotheses (4 hypotheses)

Figure 6.5 Results of the part level from the image of Figure 6.1 .

events could produce similar image events. For example, certain junction relationships between local surface patches could be due to self-occlusion of a single part or to a joint between different parts. For lack of space, we omit the details of the improvements to the part level.

Figure 6.5 shows results of the part level on the image of Figure 6.1 . All four verified parts consist of aggregrates of local surface patches (the pot for e.g. consists of two due to the dividing marking across its surface). Notice that the complete pot has been recovered although its boundary is occluded by both the spout and the flat object. This is discussed in section 6.6. In this example spurious hypotheses have been rejected at the verification stage.

6.3 The Object Level

Since finding complex objects consists of finding their parts and the relationships (joints) between them, it is also useful to analyze the generic image events between parts that allow us to hypothesize those relationships. In this section, we discuss the properties of the joints we address in this paper and outline the method used to form compound object descriptions.

6.3.1 Properties of Joints

There is a variety of ways parts can be joined in a compound object. In this paper, we consider two common types of joints: end-to-end and end-to-body (other types of joints could easily be incorporated). In the former the parts are joined such that their ends are in contact and in the latter such that one part's end is in contact with the other part's body (see Figure 6.6). The properties of these two types of joints consist of the closure patterns of the joined parts and the observed junction relationships between the joined parts' boundaries. Analyzing the different possibilities, as relate to

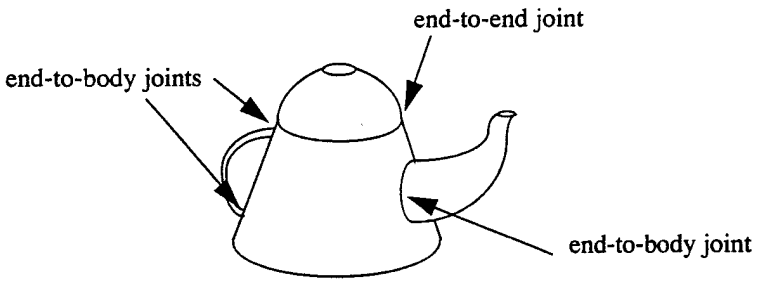


Figure 6.6 Examples of joints between parts

viewpoint for example, is useful for hypothesizing joint relationships between detected parts in the image. The properties are discussed below.

6.3.2 End-to-end joints

Our model of an end-to-end joint has two possibilities: the two cross-sections have the same size (Figure 6.7 .a through c) at their contact or have different sizes (Figure 6.7 .d and e). The observed closure patterns of the parts and the image relationships between their boundaries depend on both the parts' shape parameters (for example sweep derivatives and axis curvature) and the viewing direction (or the object's pose). In the latter case, the intersection curve (the joint curve) may or may not be visible in the image and self-occlusion may be observed. Note that the parts contact is not necessarily at their cross-sections (the joint curve may not be the cross-section curve of either part). The different arrangements for both joint closures and events between parts are shown in Figure 6.7 . The abbreviations for the junctions are as follows: *L-j* stands for *L*-junction, *T-j* for *T*-junction and *3-tgt-j* for three-tangent junction (from the catalog given in [82]).

In case *a.*, there is no self-occlusion. In case *b.* there is self-occlusion and the joint curve is visible. In case *c.* there is self-occlusion and the joint curve is not visible. Cases d. and e. have self-occlusion and differ in the visibility of the joint curve. In the figure, examples with *L-j L-j* closure could be replaced by any of the image closure patterns that result from the cross-section facing away from the viewer and the examples with *3-tgt-j 3-tgt-j* closure could be replaced by any of the image closure patterns that result from the cross-section facing toward the viewer.

6.3.3 End-to-body joints

Our model of end-to-body joint consists of a part's end in contact with another part's body. The closure patterns of the joined part and the image relationships between the parts depend on whether the joint curve is visible or not. In Figure 6.8 .a, the joined part has an *L-j L-j* closure and *T*-junctions with the other part's boundaries. In Figure 6.8 .b, the joined part has *T-j T-j* closure where the *T*-junctions are with the other part's boundaries.

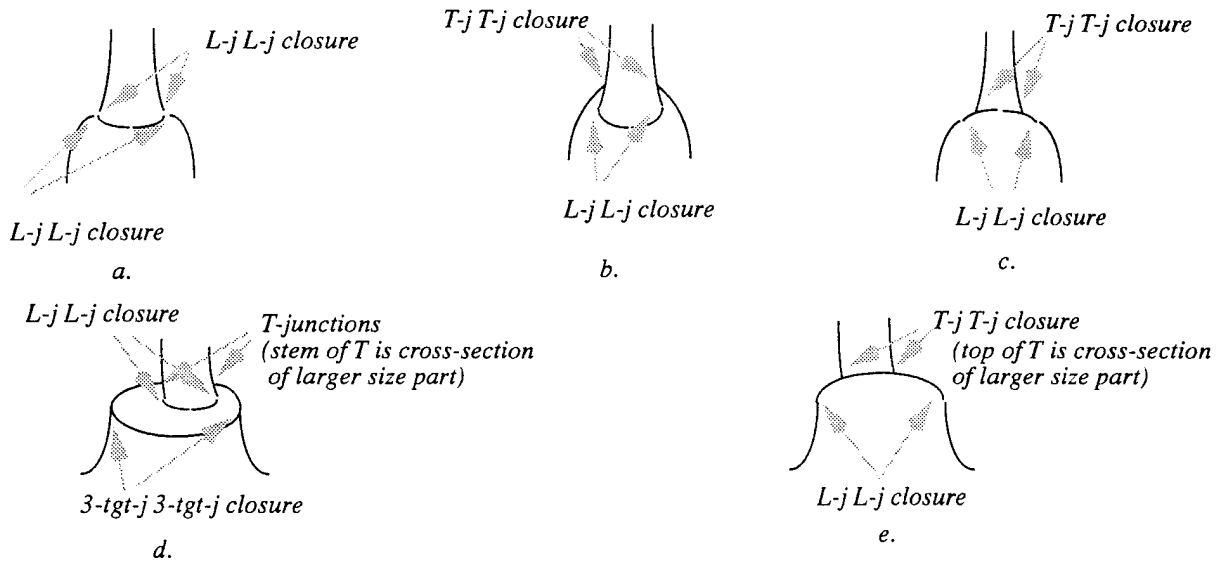


Figure 6.7 Structural relationships for end-to-end joints.
Equal size ends (a through c) and different size ends (d and e).

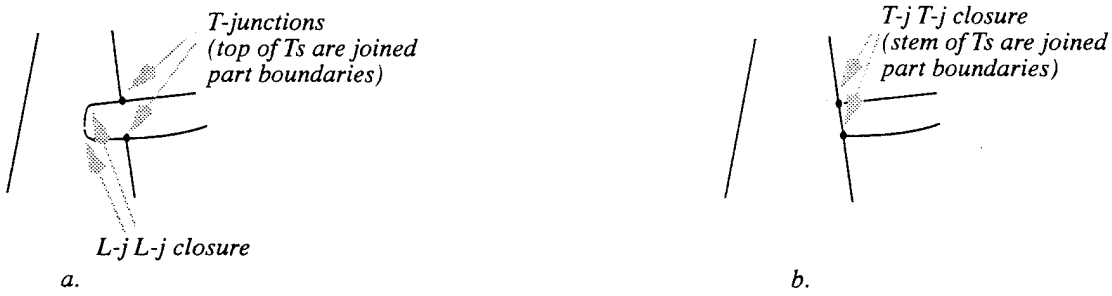


Figure 6.8 Structural relationships for end-to-body joints.
a. visible joint curve.b. non-visible joint curve

The above joint models allow for partial occlusion. Effects of occlusion by other bodies and of contour breaks are discussed in section 6.4.

6.3.4 Detection of Compound Objects

Detection of compound objects consist of hypothesizing joint relationships between detected parts. The process is not as direct as simply detecting the joint properties given previously. An inherent issue to monocular analysis of 3-D scenes is the ambiguity of the projective properties (an example was given in Figure 6.2). Thus, multiple interpretations are possible from contours alone. Further, since our goal is to produce descriptions in terms of GCs and their relationships, we must also produce descriptions that are as complete as the image allows to infer. These descriptions could be 3-dimensional if sufficient information is available in the image or otherwise 2-dimensional but corresponding to the *projections of the 3-D descriptions*. This level addresses these issues. It is organized in four steps (see Figure 6.9): detection of

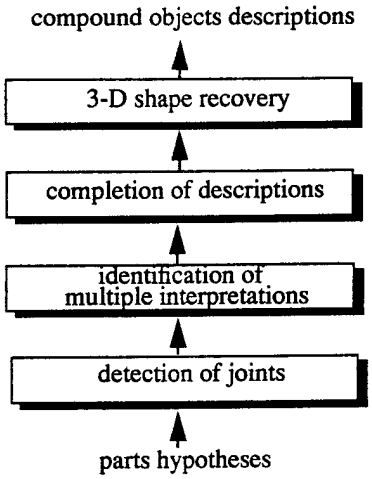


Figure 6.9 Block diagram of the object level

a. b.

Figure 6.10 Joint detection allows for partial occlusion. a. a joint is marked between parts p1 and p2 .b. no joint is marked

joints, detection of multiple interpretations, completion of description and 3-D shape recovery. We discuss these four steps below.

6.4 Detection of Joints

The objective of this step is to identify potential joint relationships between hypothesized parts. Whether there is actual (physical) contact between parts cannot be concluded from an image. The detection method consists of finding for each primitive the structural relationships of Figure 6.7 and Figure 6.8 with other primitives. Since most of those relationships involve *T*-junctions, these are first detected for all hypothesized parts (some of them are given from the analysis of the part level). The algorithm for checking any of the end-to-end or end-to-body joints between a pair of parts is fairly simple. Between a pair of parts, it uses an analysis of their contact (the same closing curve in case *a* of Figure 6.7 for e.g, or *T*-junctions between them), of the closure patterns at the joined regions and of an "extent" analysis. The latter, in case *d* of Figure 6.7 for example, consists of verifying that the closing curves of the smaller size part are all "inside" the region bounded by the cross-section boundaries of the other part's larger size cross-section. The closure constraints of the joined parts are relaxed to include occlusion at at most one side of a part's end. For example, the example of Figure 6.10 *.a* is accepted as a joint, whereas the one of Figure 6.10 *.b* is not.

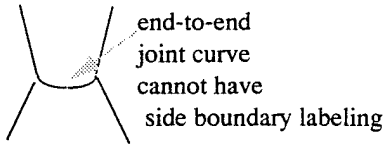


Figure 6.11 Visible joint boundaries suggest "part-end" boundary labelling

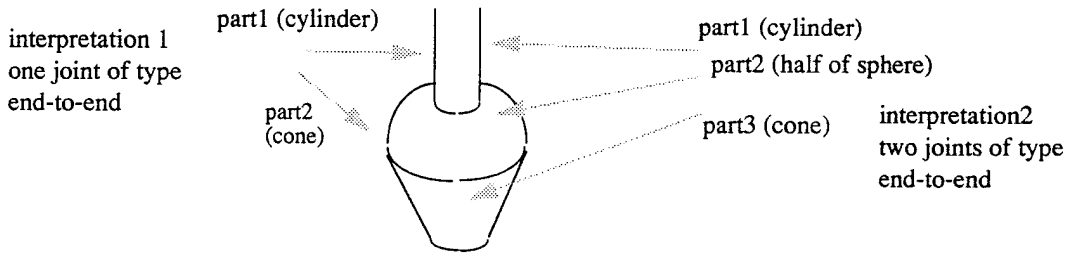


Figure 6.12 Some ambiguities persist

6.5 Identification of Multiple Interpretations

This step attempts to identify cases where more than one 3-D interpretation is possible from the given descriptions detected so far. First, the detected joints, providing global structure (or context), are used to filter out inconsistent interpretations. The joint relationships can be thought of as non-accidental relationships whose presence in the image suggests certain labelings of boundaries. The idea is that joints with visible joint boundaries (intersection curves) are unlikely to have occurred by chance in the image and they should be interpreted as parts ends. As shown in Figure 6.11 , the joint boundaries are unlikely to have side (or limb) boundary labeling (an instance of this situation was illustrated in Figure 6.2 .b). Therefore, while the single part of Figure 6.2 , taken by itself could be interpreted as either an SHGC or a PRGC, when considered in the joint of Figure 6.2 *.b*, it can only be interpreted as an SHGC part.

Remaining ambiguities are those for which certain image boundaries have different parts and joints interpretations (for e.g boundaries which could be interpreted as either cross-section or side boundaries). Figure 6.12 gives an example where two interpretations are possible: a joint of type end-to-end between two parts or two joints of type end-to-end between three parts. Therefore, conflicting parts hypotheses (two or three in this case) imply conflicting joints hypotheses (one or two joints for the same case).

The result at this stage of the method is a set of possible interpretations each of which is represented a set of graphs (one for each compound object) whose nodes are the parts and whose arcs are the joints between the parts. The arcs are labeled partially by the type of joints they represent. This graph is only a representation of the detected objects. Its purpose is not the same as the one in the method of [90] where the graph was used to segment objects made up of ribbons. Although multiple inter-

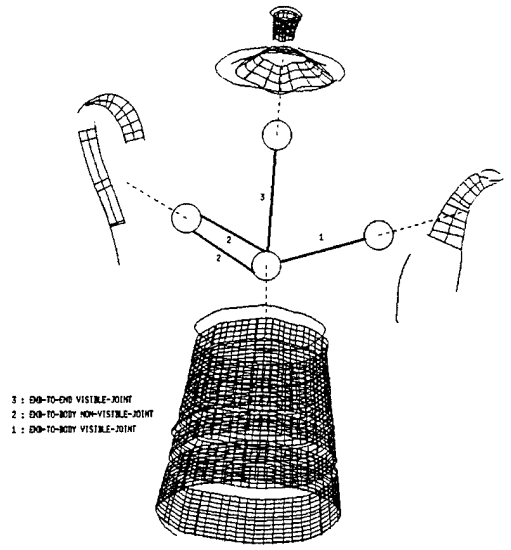


Figure 6.13 Resulting graphical representation from the hypothesized parts of Figure 6.5

pretations are a feature of our system, in the examples given in this paper, only one interpretation is found for each image. Figure 6.13 shows the graph constructed for the parts of Figure 6.5 .

6.6 Completion of Descriptions

A complete part description is one which gives its cross-section and the correspondences between its sides (projections of points on the same cross-section in 3-D), both of which give the projection of the 3-D description. Having these two elements is essential for constraining the 3-D shape of the part [93,94,95,96,97,98,99]. Some parts may already have complete projective descriptions (whole body and cross-section visible). Depending on the arrangements of parts, (self) occlusion and image contrast, some parts may miss portions of their body or their cross-sections. Two types of completions for these parts are possible: in the image and after 3-D shape is recovered.

For SHGCs with visible cross-section and partially occluded bodies, the description can be (uniquely) completed using the partial part's (projective) description [98,99]. For other parts or when the cross-section is not visible, the method attempts first to infer missing shape information, such as non-(directly)-visible cross-sections and missing side-boundaries, to the extent possible from the joint relationships between parts. For this, it is useful to classify the cut of each part as to whether it is likely to be planar or even cross-sectional. Having this classification also helps select the appropriate 3-D recovery methods [95]. To do this, we can use the geometric properties of SHGCs and PRGCs. A summary of these properties (under orthographic projection) is given below. Their use is in the reverse sense; i.e. given the observed (non-accidental) properties, we hypothesize that the part has the corresponding type of cut.

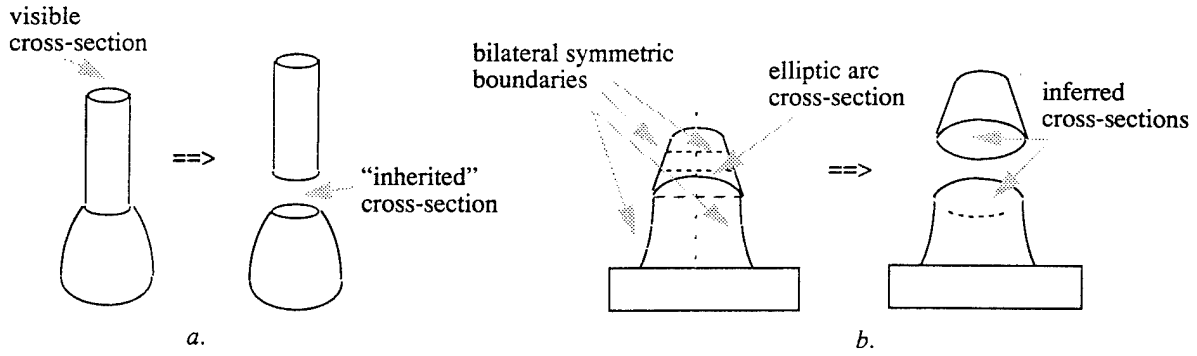


Figure 6.14 Completion of descriptions: inferring parts cross-sections

- a cross-section cut of an SHGC at both ends produces *linearly parallel symmetric*[1] boundaries in the image [93]
- a planar cut of an LSHGC at both ends produces *line-convergent*[2] symmetric boundaries in the image [95]
- a cross-sectional cut of a circular PRGC (and circular PRCGC) at any end produces an ellipse whose minor axis is *parallel* to the tangent to the projection of the axis [96]

Two types of cross-section inference are possible. One is through the use of the end-to-end joints with same-size ends (cases *a*, *b* and *c* of Figure 6.7). For example, in Figure 6.14 .a, the top part has a visible cross-section which can be "inherited" by the bottom part. This propagation can be carried through a sequence of joints. In case this is not possible, the other type of inference consists of inferring circular cross-sections for primitives which could consistently be described as such. This includes SHGCs with bilateral symmetric side boundaries [86] and circular PRGCs satisfying the third property above and having elliptic cross-sectional arcs as the partially visible cross-section. Figure 6.14 .*b* gives an example.

For parts which cannot be completed in the image and for which partial 3-D shape can be recovered, the completion is done in 3-D and consists of filling in the gaps in the 3-D axis by quadratic curves (in its recovered plane) and the gaps in the sweep function by piecewise linear sweeps.

The projective completion of the pot of Figure 6.13 (main part) has been done using only the SHGC description. The completion of the spout (right PRGC) is done after 3-D shape is recovered (see Figure 6.15). The handle (left PRGC) could not be completed since its cross-sections and joints are not visible due to occlusion.

---

1. Parallel symmetry is a generalization of parallelism of straight lines to curved ones. It is linear if the curves are scaled and translated version of each other [95].
2. Line-convergence is a form of symmetry whereby tangent lines at symmetric points of two curves intersect along a line[95]. It can be thought of as the generalization of point incidence to line incidence.

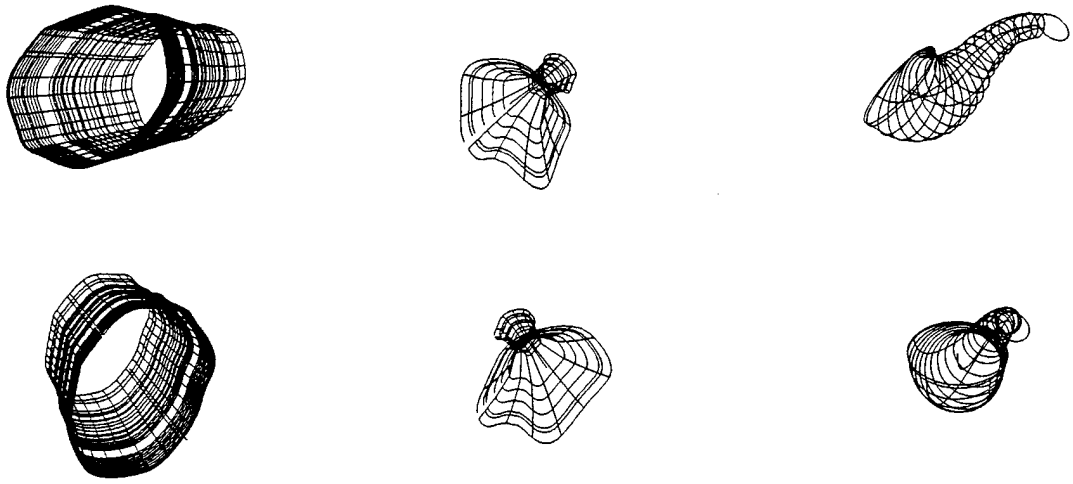


Figure 6.15 Recovered 3-D volumetric descriptions for the descriptions of Figure 6.13 .

6.7 3-D Shape Recovery

At this stage of the system, possible interpretations of image boundaries in terms of compound objects are identified. Recovering 3-D shape of a compound object consists of recovering the intrinsic 3-D description of each of its parts; i.e. its 3-D cross-section, its 3-D axis and the sweep function. 3-D descriptions from monocular images of SHGCs, PRCGCs and circular PRGCs have been addressed by the authors and others in [76,80,85,89,93,98,97], [94] and [96] respectively. Those method are based on using the properties of a primitive to generate constraints on its 3-D shape. For compound objects, a complete 3-D recovery method should normally use, besides such properties for each part, constraints on the interaction between parts. For example, well defined differential geometric relationships hold between the orientations of two surfaces and their intersection curve [78]. We have not attempted to address this problem in our work though (this is a topic in its own right).

We have instead used the methods described in [93,98,97,96] to recover 3-D shape of each part as though it were isolated. Figure 6.15 shows the recovered volumetric descriptions of the parts of the object in Figure 6.13 , using different poses in 3-D (the descriptions are shown in terms of cross-sections and meridians of the recovered 3-D GCs). Notice the 3-D completed spout. The handle could not be recovered since its cross-sections are not (even partially) visible (its description remains projective)

Figure 6.16 shows results of the method on another image. Both mugs consist of a main body (SHGC) and a handle (PRGC). Notice the markings on the surface of the cup in the front and outside the objects. The image also exhibits occlusion between independent objects. There were 177 initial local surface patch hypotheses (Figure 6.16 .c) and two objects, each made up of two parts joined by two joints, are ob-

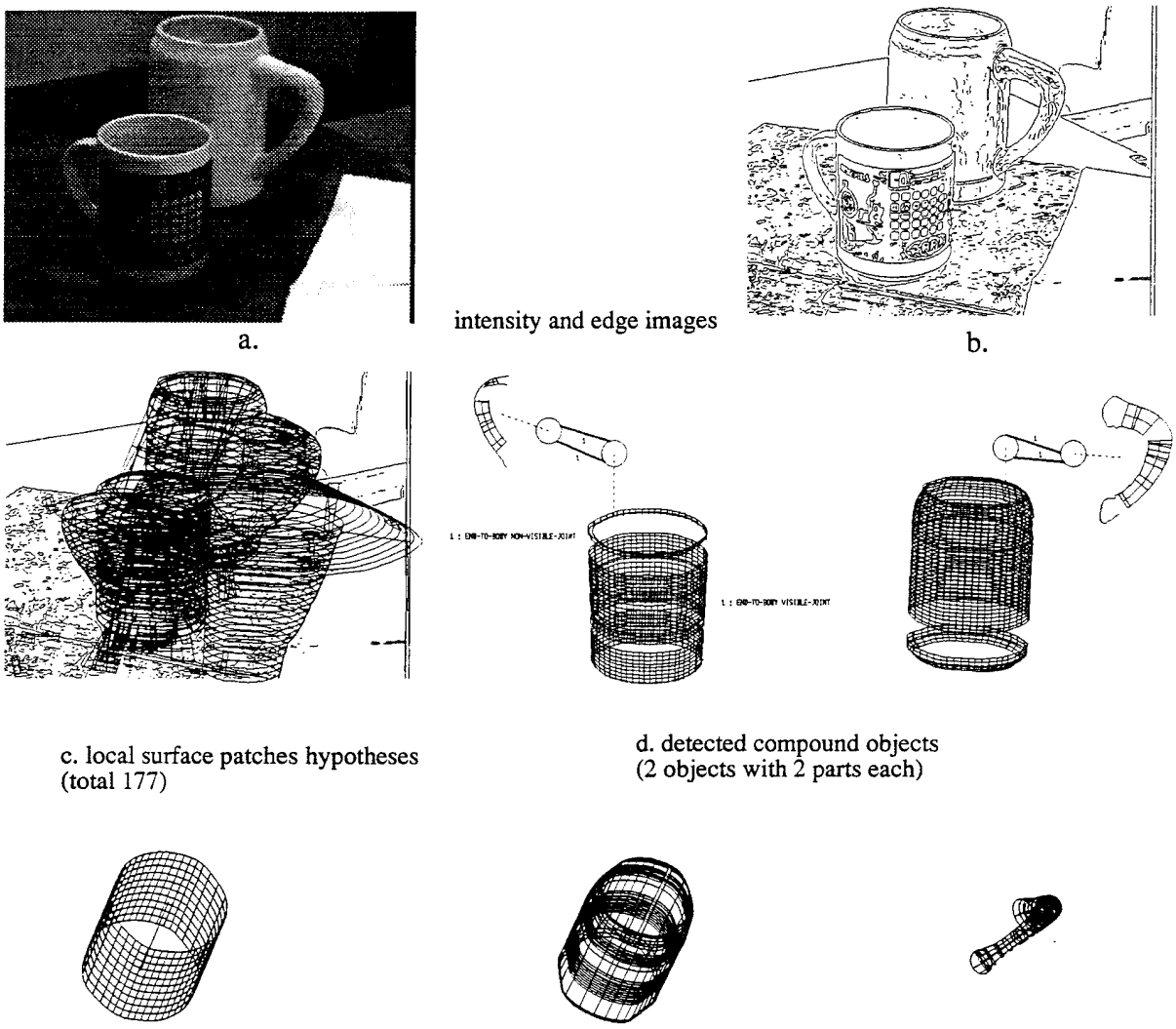


Figure 6.16 Additional results of the method

tained. The joints labeling and the obtained graphical representations are shown in Figure 6.16 .d. The recovered 3-D parts are shown in Figure 6.16 .e for different orientations. The 3-D shape of the handle of the front mug could not be recovered because its cross-sections are not visible. At this stage, the 3-D parts and their relationships are completely identified.

6.8 Discussion and Conclusion

There are a number of issues not addressed in this paper. Among them is handling parts with multiple surface patches such as occurs with polyhedral cross-section parts. This issue has been partially resolved in a previous work in the case of concave cross-section SHGCs [98,99]. To handle this type of parts requires a further step

whereby surface patches generated from different portions of the cross-section are identified and merged.

In summary, the proposed method makes use of the projective properties of a small number of primitive GCs and their relationships in order to recover segmented 3-D descriptions independently of the viewing direction and in the presence of partial occlusion, surface markings, shadows and contour breaks.

The results of this work have several applications. The descriptions obtained by our system (either the 3-D intrinsic elements of a GC or their projective descriptions) can be used to provide powerful, view-insensitive, indexing keys to large databases of object models for object recognition (such as in [87] for example). In manipulation, the 3-D descriptions can be used to plan for the grasp and pre-shape the hand. In navigation, they can be used to select appropriate paths to avoid obstacles, for example, and in learning, the symbolic descriptions can be used to analyze differences and similarities between newly recovered objects and previously recovered ones.

6.9 References for Chapter

[71] H.G. Barrow and J.M. Tenenbaum, "Interpreting Line Drawings as Three Dimensional Surfaces" Artificial Intelligence, 17:75-116, 1981.

[72] R. Bergevin and M.D. Levine, "Generic Object Recognition: Building and Matching Coarse Descriptions from Line Drawings," in *IEEE Transactions PAMI*, 15, pages 19-36, 1993.

[73] P.J. Besl and R.C. Jain, "Segmentation Through Symbolic Surface Descriptions," In *Proceedings of IEEE CVPR*, pages 77-85, 1986.

[74] I. Biederman, "Recognition by Components: A Theory of Human Image Understanding," *Psychological Review*, 94(2):115-147.

[75] T.O. Binford, "Visual Perception by Computer," *IEEE Conference on Systems and Controls*, December 1971, Miami.

[76] M. Dhome, R. Glachet and J.T. Lapreste, "Recovering the Scaling Function of a SHGC from a Single Perspective View," In *Proceedings of IEEE CVPR*, pages 36-41, 1992.

[77] S. Dickinson, "3-D shape Recovery using Distributed Aspect Matching," *IEEE Transactions PAMI*, 14(2):174-198, 1992.

[78] Do Carmo, "Differential Geometry of Curves and Surfaces," Prentice Hall, 1976.

[79] T.J. Fan, G. Medioni and R. Nevatia, "Recognizing 3-D Objects using Surface Descriptions," *IEEE Transactions PAMI*, 11(11):1140-1157, 1989.

[80] A. Gross and T. Boult, "Recovery of Generalized Cylinders from a Single Intensity View," In *Proc. Image Understanding Workshop*, pages 557-564, 1990.

[81] J.E. Hummel and I. Biederman, "Dynamic Binding in a Neural Network for Shape Recognition" *Psychological Review*, 1992.

[82] J. Malik,"Interpreting line drawings of curved objects," *International Journal of Computer Vision*, 1(1):73-103, 1987.

[83] D. Marr, "Vision," W.H. Freeman and Co. Publishers, 1981

[84] R. Mohan and R. Nevatia, "Perceptual organization for scene segmentation," *IEEE Transactions PAMI*. 1992.

[85] T. Nakamura, M. Asada and Y. Shirai, "A qualitative approach to quantitative recovery of SHGC's shape and pose from shading and contour," In *Proceedings of IEEE CVPR,* pages 116-121, New York, 1993.

[86] V. Nalwa, "Line drawing interpretation: Bilateral symmetry," *IEEE Transactions PAMI,* 11:1117-1120, 1989.

[87] R. Nevatia and T.O. Binford, "Description and recognition of complex curved objects," *Artificial Intelligence*, 8(1):77-98, 1977.

[88] A. Pentland, "Recognition by Parts," in *Proceedings of the ICCV*, pages 612-620, 1987.

[89] J. Ponce, D Chelberg and W.B. Mann, "Invariant properties of straight homogeneous generalized cylinders and their contours," *IEEE Transactions PAMI*, 11(9):951-966, 1989.

[90] K. Rao and R. Nevatia, "Description of complex objects from incomplete and imperfect data," In *Proceedings of the Image Understanding Workshop*, pages 399-414, Palo Alto, California, May 1989.

[91] M. Cecilia Rivara, "Algorithms for Refining Triangular Grids Suitable for Adaptive and Multigrid Techniques," *International Journal for Numerical Methods in Engineering*, Vol. 20, pp. 745-756, 1984.

[92] L. Roberts, "Machine Perception of Three-Dimensional Solids," MIT Press, 1965.

[93] F. Ulupinar and R. Nevatia, "Shape from contours: SHGCs," In *Proceedings of ICCV*, pages 582-582, Osaka, Japan, 1990.

[94] F. Ulupinar and R. Nevatia, "Recovering Shape from Contour for Constant Cross Section Generalized Cylinders," In *Proceedings of Computer Vision and Pattern Recognition*, pages 674-676. 1991. Maui, Hawaii.

[95] F. Ulupinar and R. Nevatia, "Perception of 3-D surfaces from 2-D contours," *IEEE Transactions PAMI*, pages 3-18, 15, 1993.

[96] M. Zerroug and R. Nevatia, "Quasi-invariant properties and 3D shape recovery of non-straight, non-constant generalized cylinders," In *Proceedings of IEEE CVPR,* pages 96-103, New York, 1993.

[97] M. Zerroug and R. Nevatia, "Using invariance and quasi-invariance for the segmentation and recovery of curved objects," in *Proceedings of the 2nd ARPA/ESPRIT Workshop on Geometric Invariance in Computer Vision*, The Azores, 1993.

[98] M. Zerroug and R. Nevatia, "Segmentation and 3-D recovery of SHGCs from a single intensity image," to appear in *European Conference on Computer Vision*, Stockholm, 1994.

[99] M. Zerroug and R. Nevatia, "Volumetric descriptions from a single intensity image," to appear in *International Journal of Computer Vision*.

7 Extraction of Groups for Recognition

Parag Havaldar, Gérard Medioni and Fridtjof Stein

We address the problem of recognition of generic objects from a single intensity image. This precludes the use of purely geometric methods which assume that models are geometrically and precisely designed. Instead, we propose to use descriptions in terms of features and their qualitative geometric relationships. We propose to detect groups using perceptual organization criteria such as proximity, symmetry, parallelism, and closure. The detection of these features is performed in an efficient way using proximity indexing. Since many groups are created, we also perform selection of relevant groups by organizing them into sets of similar perceptual content. Finally we present an initial implementation of a recognition system using these sets as primitives. It is an efficient colored graph matching algorithm using the adjacency matrix representation of a graph. Using indexing, we retrieve matching hypotheses, which are verified against each other with respect to topological constraints. Groups of consistent hypotheses represent detected model instances in a scene. The complete system is illustrated on real images. We also discuss further extensions.

7.1 Introduction

Most object recognition systems today address the problem of finding the location and orientation of an exactly known rigid object in a scene. Grimson's book [109] gives a lucid treatment for the *geometric constraints* used in these approaches. The presence of a model is inferred by the verification that such a model could indeed produce some of the observed data under an appropriate geometric transform. However, this approach cannot be extended to more general scenarios containing objects which may be very similar while being geometrically different. Consider for instance two different airplanes which have similar features but different geometries. In other words, generic recognition obviates the use of methods based purely on the exact geometric structure of the object. It is clear that the only way to solve this difficult problem is to reason about parts and their arrangements. This argument is supported by Biederman's theory [101], which states the sufficiency of a limited number of volumetric components (or geons) for the task of recognition. Recovery of parts and their arrangements can help fast recognition of objects even if they are occluded, novel, rotated in depth or extensively degraded.

We therefore have three problems to solve: the extraction of primitives, the description of scenes in terms of these primitives and the actual recognition of objects.In this paper we propose the use of perceptual grouping to approach the problem of generic recognition. Use of perceptual groups is not new, as it was proposed in the 1970s

but was not very successful because of the failure to obtain reliable primitives in the first place. Using groups explicitly for recognition was first launched by the classic work of L. G. Roberts [118]. Brooks [103]developed an image understanding system called ACRONYM which uses a restricted class of generalized cylinders (GC) for descriptions of model and scene objects. Lowe's SCERPO[113] system takes a bottom-up approach to object-centered recognition. Rothwell in [120] explains the need for computing local invariants and for tying them together to form complete object descriptors as opposed to computing a single global descriptor.

In section 7.2 we describe the feature hierarchy computed. As explained in some of our previous work [124], such groups serve as an intermediate level representation of the data, in a hierarchical fashion, and can be used to retrieve likely candidate objects from a library. Some of the groups extracted may not yield any natural descriptions. Hence, in section 7.3, we perform a selection step by organizing the groups into sets which have similar "perceptual" content making use of junctions to reason about relevant sets. In section 7.4, we give an outline of our recognition system which uses these sets for recognition. As models we use multiple views of an object. Results are shown in section 7.5.

7.2 Going from Edgels to Groups

In our previous work [122], the super segment was introduced in two or three dimensions as a feature to represent a piece of a curve. It is based on the assumption that the underlying structure embodies continuity. Here, we propose to go to higher level groups which take into account other grouping criteria besides co-curvilinearity, such as parallelism, closure, and symmetry. In computer vision, many authors have focused on computing perceptual groups (see *e.g.* [110,113,114,111,119,121,127]). Most of these algorithms tackle the detection of *all* perceptual groups by either assuming perfect data, or by applying exhaustive search. Our algorithms try to compromise: we do not assume perfect data and therefore we find most (but not necessarily all) perceptual groups. On the other hand, we do this in an efficient way by using *proximity indexing*.We now explain the steps involved in going from an image to a high level representation of it in terms of "perceptual" groups.This chain of processing is sketched in Figure 7.1 .The following sections focus on the details of the perceptual hierarchy.

7.2.1 Preprocessing

In the *preprocessing* stage we reduce the amount of data: starting from images we first detect edges in the image (using the Canny edge detector [104]). We then compute curves, which consist of linked edgels. In the *local grouping* stage we generate many line segments based on multiple linear approximations with different fitting tolerances. For each approximation tolerance, we perform a vertex collapse and compute super segments and parallels. By creating a large set of features at this point we gain robustness in our further groups, and we significantly reduce the unreliability of

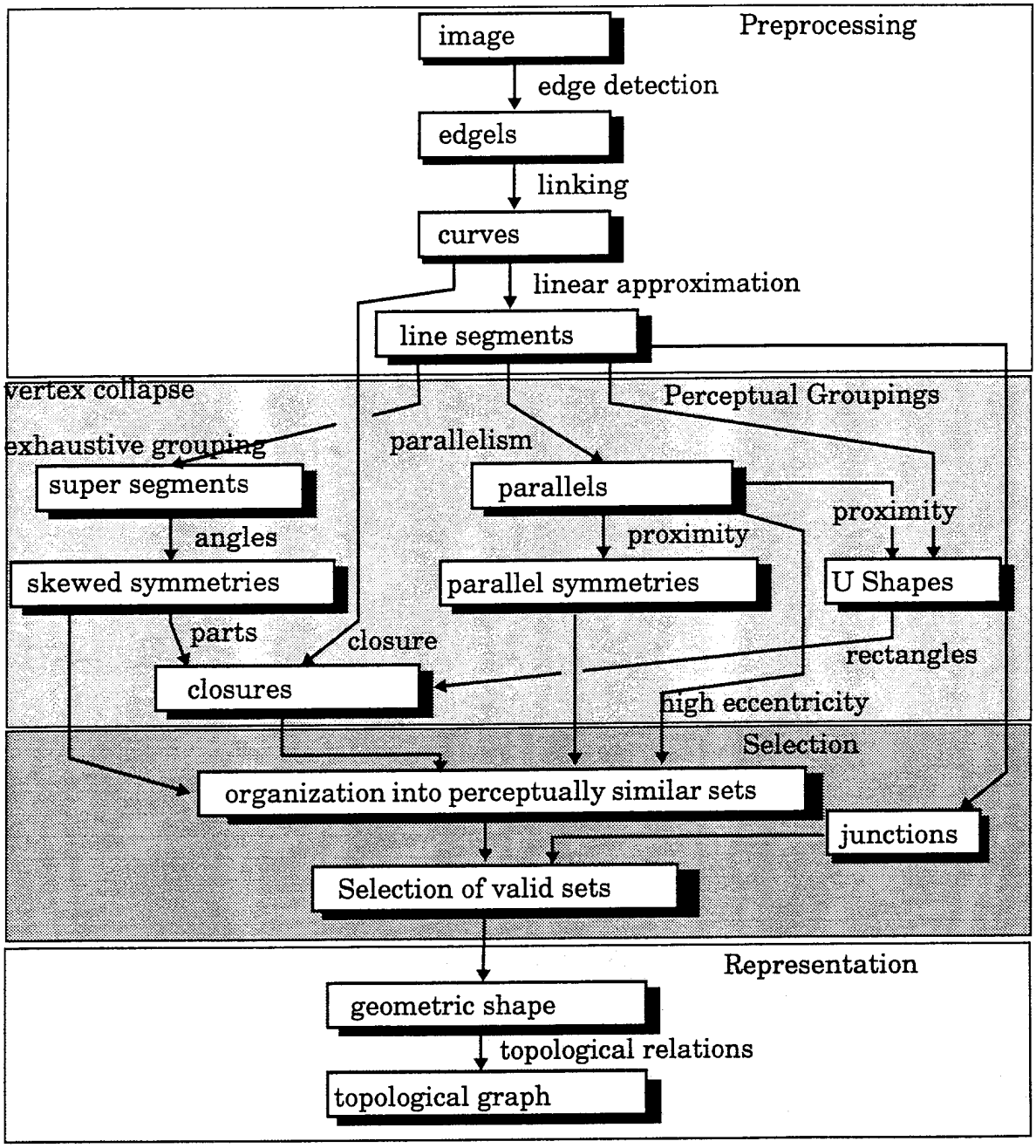


Figure 7.1 Feature Hierarchy

the preprocessing. The *perceptual grouping* stage no longer distinguishes between features of different fitting tolerances. Below in Figure 7.2 we show an example scene and the detected edges. Note that although this is the same image as the one used in Zerroug and Nevatia [128], we follow a very different line of reasoning. In particular, we make no assumptions about the kind of objects we deal with.

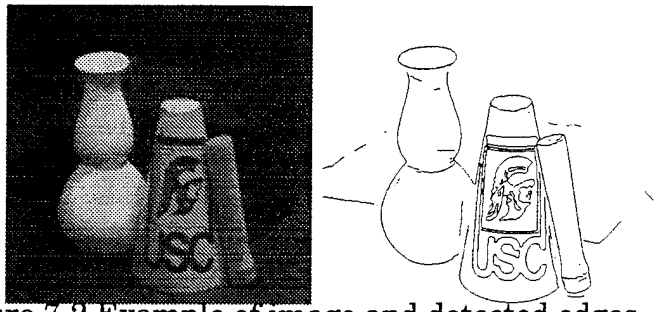


Figure 7.2 Example of image and detected edges

7.2.2 Super Segments

Since we want to handle occlusion, we do not expect to obtain complete boundaries in our images, but only portions of them. Grouping a fixed number of adjacent segments provides us with one of our basic features, the super segment. The computation of super segments is the same as described in [122]. Connected linear segments form chains of adjacent segments. We generate super segments from cardinalities 3 to 6.

7.2.3 Parallels

Segments which are parallel within a certain tolerance (δ=20°) are grouped as parallels. For each linear segment, the possible candidate parallels are retrieved and verified with respect to aspect ratio and overlap. Segment pairs which meet theses constraints generate parallels. Using proximity indexing, we are guaranteed to find parallels which are at most $\delta/2$ apart and we get some parallels with angles between $\delta/2$ and δ.

7.2.4 Symmetries

Symmetries have been used by various authors [114,127]. We detect two specific symmetries here as features of an object.

Parallel symmetries are retrieved by finding proximate parallels. We do not use the super segment approach, because we would depend on the cardinality of the super segments. By using the parallels as the building blocks, we can use proximity indexing to find parallels which share the same vertices. Examples are shown in Figure 7.3 (left). Skew symmetry was first proposed by Kanade [110] and its extraction was done by Ponce [111] and Saint-Marc and Medioni [121] but these methods are quite sensitive to noise. We are interested in symmetries between line segments. Our approach is not exhaustive. We use supersegments to detect skew symmetries. Two super segments are skew symmetric, if they satisfy the following: (a) the difference between the corresponding angles must be smaller than $2\theta_{max}$, and (b) the symmetry axis has to be straight. An example is given in Figure 7.3 (middle).

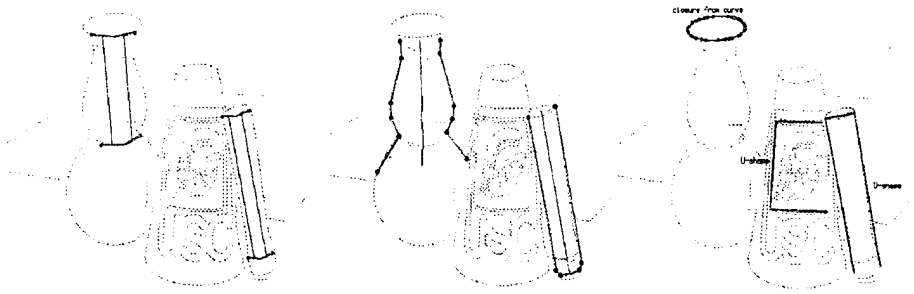


Figure 7.3 Examples of groups - parallel symmetries (left), skew symmetries (middle) and U-shapes (right)

7.2.5 Closures

Lowe [113] states: *There is a tendency for curves to be completed so that they form enclosed regions.* Based on this statement, Mohan and Nevatia [114] developed the idea to close symmetries at their ends to obtain so called *ribbons*, which form enclosed regions. They use these ribbons to segment images. We want to use closures as features. At the moment we compute closures from U-Shapes, from closed curves and from skewed symmetries. A parallel which is closed at one side by a linear segment is a strong indication that a rectangular structure is at hand where one side could not be detected. We therefore assume that we found a closed contour. U-Shapes can be found by indexing over the vertex pairs of parallels and trying to find a segment which forms a U-Shape with the parallel.The obvious form of a closure occurs if we have a closed curve. To detect a closure based on a curve we allow the gap between start and end of the curve to be 5% of the arc length of the curve. We adopt the idea that a segmentation into parts should be done at negative minima of curvature from Rom and Medioni [119]. Such "a part" is used in our system as a closure. Whenever we encounter a sign change of consecutive angles, we "break" the symmetry at this point. Applying this step iteratively, we generate alternating convex and concave parts. We use the convex parts to create closures. Examples of closures can be seen in Figure 7.3 (right).

7.2.6 Efficient Implementation through Proximity Indexing

Proximity indexing was used to efficiently compute the feature hierarchy described in the previous section. Proximity indexing issues play an important role when we wish to find features with similar attribute values. Traditional search methods which compare every possible pair are very time consuming. Recent vision systems have used indexing. A major problem with indexing is deciding the length of the quantization intervals. Values which are close, may fall in different quantization intervals as shown in Figure 7.4 . Two features match only in the case when they fall into the same interval, which may not always be so. Flynn and Jain [108] point out, it is essential to have an indexing scheme that preserves proximity in the key values. So far, two strategies based on indexing have been used to deal with this problem:

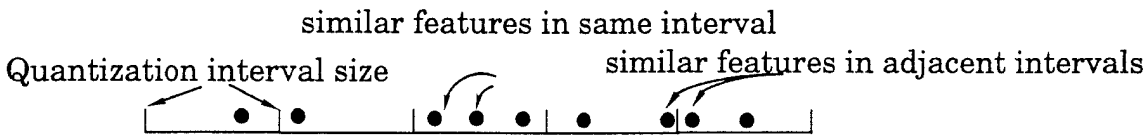


Figure 7.4 The indexing problem

large bucket size and searching of neighboring bins. While large bucket size is based on the hope that "less values will fall into the incorrect bin", the search of neighboring bins has an exponential complexity with respect to the number of false value matches.

We propose an alternative approach: We encode every feature twice and use indexing on every value separately. Every value is quantized twice. The stored features for both intervals are retrieved and combined into one set. For all values we get such a feature set. The intersection of all these sets results in the features which are close to *f*. Such an interlaced quantization is guaranteed to preserve proximity while indexing

7.3 Selection of Relevant Groups

The groups extracted from images not only contain perceptually salient features, but also contain many features which do not yield any natural descriptions. Such groups come about when the segments and supersegments give rise to features (symmetries, U-shapes etc.) which are geometrically correct, but are less obvious because of other competing groups. The undesired groups also increase the complexity of the representation and matching process (see section 7.4). Selecting relevant groups may be helped by purely local heuristics, such as the skewness or orientation of overlapping groups. We prefer to make use of more global constraints. We first aggregate the groups into different sets such that each set contains groups which are perceptually similar. Note that each set may be perceptually correct or incorrect. To pick out relevant groups we make use of junctions and reason at the level of the above sets.

The main advantage of this is that if a certain set of features can be verified as not coming from any surface of an object in the image, then the entire set may be discarded. Reasoning at such a surface patch level provides a stronger grip on the entire selection process. We now explain the aggregation and selection processes.

7.3.1 Aggregation of Groups into Sets:

We use the properties of the axis to organize the groups into various sets, the objective being that each set should perceptually provide the same information. Groups which are "similar" have similar axes orientation and their axes are spatially close together.

1) We first aggregate the groups having similar axes orientations into different sets. The groups in each set may still be spatially located at different positions.

2) Next we choose each set and further partition it depending upon spatial neighborhoods. This is done by constructing a graph whose nodes represent groups and edges represent spatial closeness. The connected components of the graph give sets of groups having similar orientation and position

3) Lastly we look at the segments of the groups in each set and further separate out groups which vary markedly in their average distance to the axis.

In Figure 7.5 we show examples of four such group sets. Each image displays one group in each set. Below we discuss how to select valid sets.

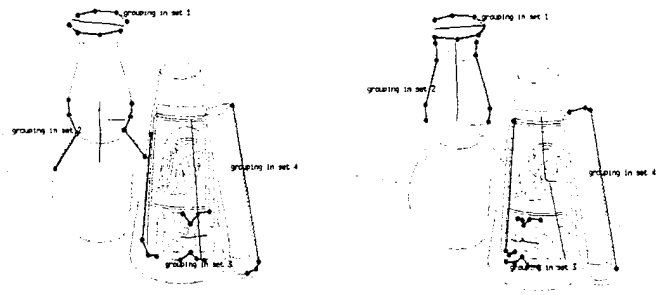
7.3.2 Selection of sets

We first compute all the junctions in the image and use them to decide the validity of the sets. The junctions in the image may be because of a variety of reasons, mainly due to occlusion, surface markings or surface-orientation discontinuities on the object. We are not trying to classify the junctions, but rather use them as a tool for verification of the sets. Some computed junctions are shown in Figure 7.5 (b). Next we break the segments of the groups in each set into two subsets, each subset containing segments on either side of the axes. If there exist junctions which connect the above mentioned segment subsets, such that one junction connects the segments of one side of the set to another set and a second junction connects the segments of the other side of the same set to a third set then we label this set as invalid. In Figure 7.5 (c) we show some of the valid and invalid groups in the set as a result of this reasoning for the junctions shown. It can be seen that the groups in set 4 are invalid. Two junctions which the segments of this group set form and meet the above reasoning are junctions 3 and 4. Another example of an invalid set comes about because of junctions 6 and 7. On the other hand junctions 1 and 2 are formed between segments of group sets 1 and 2 shown in Figure 7.5 (a). It can be seen that in this case the segments of the junctions are shared between two sets and not three.

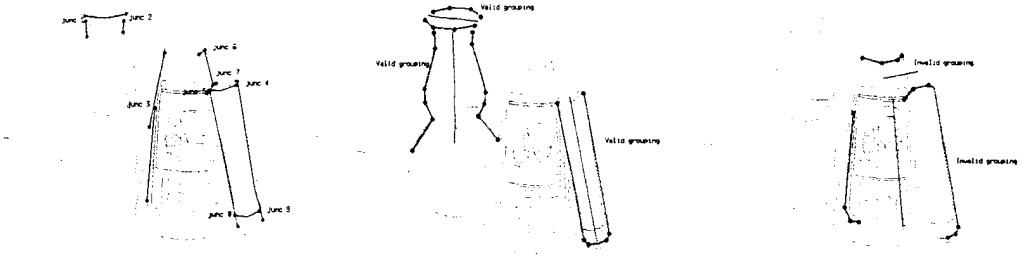
At this stage we have the groups organized into sets, each set contains groups which as perceptually similar and can now be used as tokens for recognition. Note that each set yields a multiple representation of a "part" in the form of many groups. We can encode the spatial relationships of these sets to form a graph to describe the object. Note that although the percentage of the irrelevant groups has decreased, they may not necessarily be totally eliminated. However, recognition is achieved by hypotheses voting, which tends to minimize the effect of irrelevant groups in the scene.

7.4 Representation and Matching

Given a set of features and the topological relationships between them, a natural representation of this structure is a graph. For basic graph related definitions used here and more on graph theory see [116],[100],[105]. For our topological graph the vertices are the shapes or enclosures of sets of groups and the edges represent the topological relationship between them. We compute the enclosures or convex approxi-



(a) Examples of groupings organized into sets of similar perceptual content. Four sets are shown. Each image shows one group of each set.



(b) Examples of some junctions detected in image

(c) Examples of valid and invalid groups for T-junctions shown above

Figure 7.5 Selection of Sets

mations (CAs) of the set. This gives us the geometric shape of the set which enables easy computation of the topological and spatial relationships. In the current implementation we label the edges with only two labels: adjacency and inclusion. The direction of an edge depends on its label. When an edge is labeled with "inclusion", the edge is directed from the inner set to the outer set. When an edge is labeled with "adjacency", the edge is undirected. In Figure 6.6 , we show an example of a scene of four CAs and the corresponding graph.

In computer vision, graph matching is widely used (see *e.g.* [115], [107], [114], [117]). In the worst case, the subgraph isomorphism problem is NP-complete. Therefore several heuristics were developed to improve the average complexity for specific cases (see *e.g.* [105]). Despite all the previous research, we could not find any algorithm which would perform with reasonable complexity for graphs consisting of a hundred vertices or more. Furthermore, we are unaware of theoretical results on subgraph isomorphism in colored graphs. Our goal is to find large subgraph isomorphisms, which are likely to represent detected models in a scene. We believe that we can use structural indexing to find corresponding subgraphs

We need to decide upon structural tokens to perform indexing. Using the sets of groups only would be very expensive. In our implementation the "color" of the graph lies in relationships between the sets, i.e. the edges in the graph. *We make use of the information of the groups in the sets to separate out consistent hypotheses after the can-*

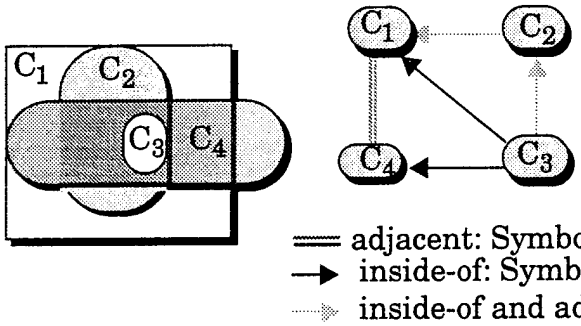


Table 1:

	C 1	C 2	C 3	C 4
C 1				a

Figure 6.6 Example of Graph

didate hypotheses have been detected. The idea is to find all paths of length k (corresponding to k+1 connected sets) and to cluster these to find the largest consistent group of such paths. In a complete connected graph consisting of $|V|=m$ vertices, there are m-1 outgoing edges at every vertex ($|E| = (|V|-1)!$). Therefore the number of paths of length k is $m(m-1)(m-2)...(m-k)$. This corresponds to an upper bound of $O(|V|^{k+1})$ in the number of paths. A more realistic assumption is that there are only a constant number of outgoing edges at every vertex. This results in an upper bound of $O(|V|^{k})$ in the number of paths. On one hand we are interested in using long paths to be as discriminative as possible, on the other hand the number of possible paths in a graph grows exponentially with k. Another consideration for k is the size of the corresponding subgraphs. Choosing a large k can result in not detecting a subgraph which has less vertices than k. In our implementation, we use the path length k=2. This allows us to exploit the discriminative power of three connected sets. At the same time the number of paths has the worst case complexity of $O(|V|^{2})$ (assuming a constant number of outgoing edges at every vertex). The clustering of corresponding paths enables us to find the corresponding subgraphs with more than 3 vertices.

The computation of the paths is straightforward. The graph can be represented by its adjacency matrix.The representation of an object works as illustrated in Figure 7.6 . Every view of a model is processed in the following way:

1) The feature hierarchy is computed.
2) The enclosures of the sets are used to create the topological graph.
3) All paths (in our implementation of length 2) are computed.
4) Each path is encoded
5) Each path is stored in a data base.

To encode a path we take the code of all pairs of sets. We use the following attributes to encode a pair of sets:

- the label of the connecting edge,
- when the two sets are adjacent, the percentage of common boundary,
- when the two sets are intersecting, the percentage of area intersection.

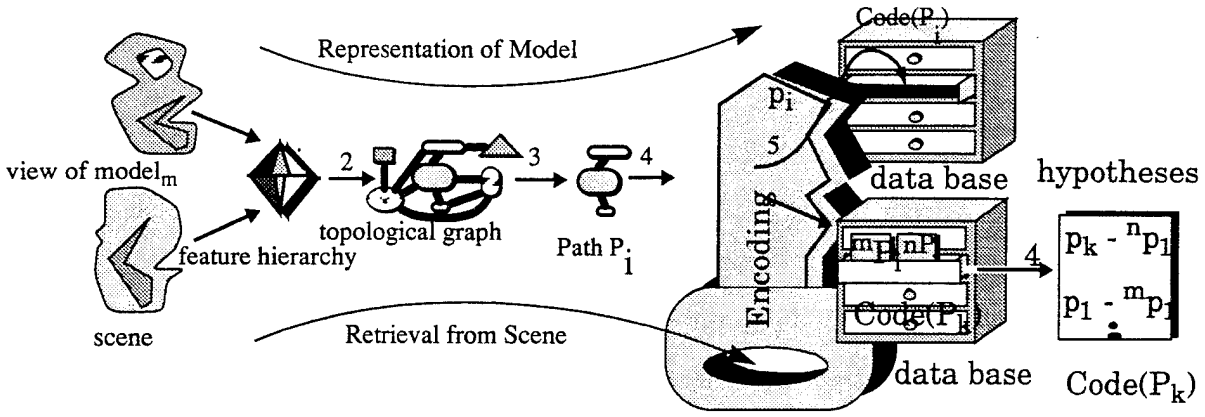


Figure 7.6 Representation of Model and Scene

All numerical values are quantized in a coarse way to allow significant deviations due to viewpoint change and noise. The typical quantization in our implementation for all numerical values is 20%.

The generation of the hypotheses proceeds in a similar way (see Figure 7.6)

1) we perform steps 1 through 4 above, then we
2) retrieve from the data base the stored model paths which have the equivalent code.

The retrieved hypotheses are equivalent to subgraph isomorphisms of path length 2 between the different model-views and the scene. In the verification step we cluster the hypotheses in order to get larger corresponding subgraphs which are likely to represent an instance of a model in a scene. Two hypotheses are consistent when the following rules apply:

1) They share at least one corresponding set pair.
2) No contradiction occurs. That means that the combined number of vertices and edges of the two paths in the model-view have to be the same as in the scene.
3) Connectivity has to be preserved. When two vertices are connected in one subgraph they have to be connected with the same label in the corresponding subgraph.

In this case, the combined hypotheses form a new hypothesis. These clusters grow iteratively until no further consistent hypotheses pairs can be found.

7.4.1 Analysis

What would have happened if we would have taken shorter or longer paths as basic matching primitives? Given k as the path length. Then c_k is the number of CA pairs in a path consisting of k+1 vertices, with $c_k=(k+1)k/2$. In section 7.4 we talked

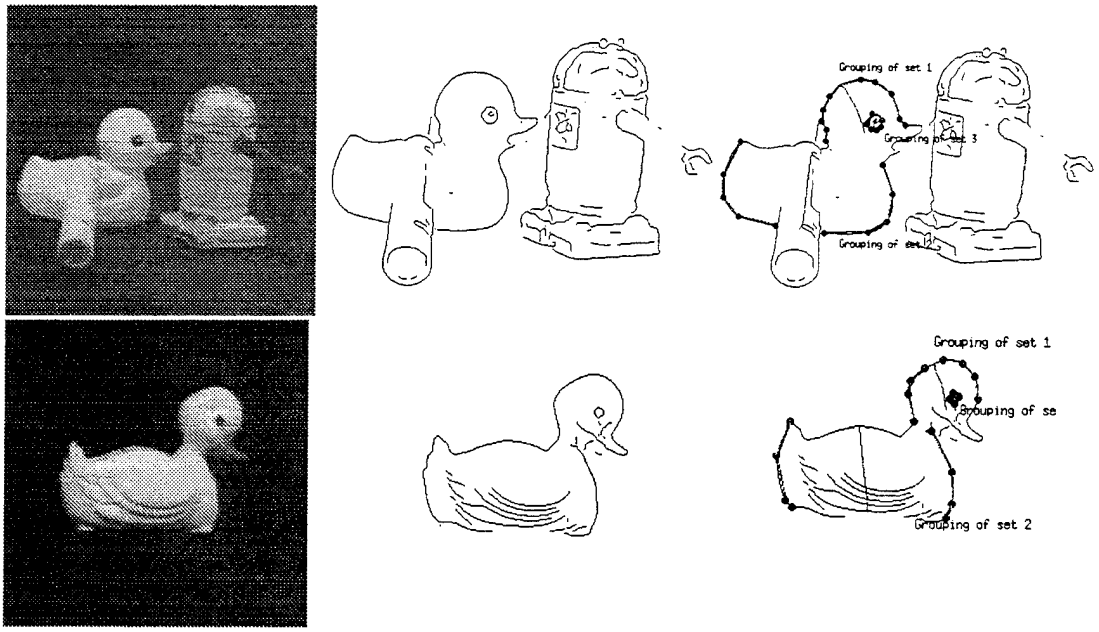
about the attributes which we use to encode a pair of enclosures of sets: the label of the connecting edge, the percentage of common boundary, and the percentage of area intersection. The label can have four different values: nil, adjacent, inside, or adjacent and inside. We further mentioned that we quantize the last two values in five quantizations of 20% each. That means we have a=4*5*5=100 different codes to encode a pair of sets. Therefore the number of available path codes of length k is $D_k=a^{c_k}=a^{(k+1)k/2}$.

D_k is a measure for the discriminative power of an encoding scheme. The larger D_k, the larger the code alphabet, the more discriminative power a feature has.The trade-off lies in the generation of the matching primitives versus the generation of the hypotheses with respect to the discriminative power. Taking a short path length results in a low number of paths to generate. For k=1, the number of paths is $O(|V|)$. On the other hand the discriminative power of these paths is lower. For k=1, $D_1=100^2=10000$. Because the number of paths decreased by one order of magnitude and the discriminative power decreased by two orders of magnitude, the number of generated hypotheses h is in general larger than in our example. Because the clustering of the hypotheses has a complexity of $O(h^2)$, the matching and verification for k=1 is slower than for k=2. Taking a long path length results in a large number of paths. For example, for k=4, the number of paths is $O(|V|^4)$, and $D_4=100^{10}=10^{20}$. The number of generated hypotheses will be minimal due to such a high discriminative power, and therefore the final clustering will be very fast. But computing $O(|V|^4)$ paths requires a high space complexity which may be prohibitive. The right way to proceed is to increase the value a. This can be done by improving the encoding scheme.

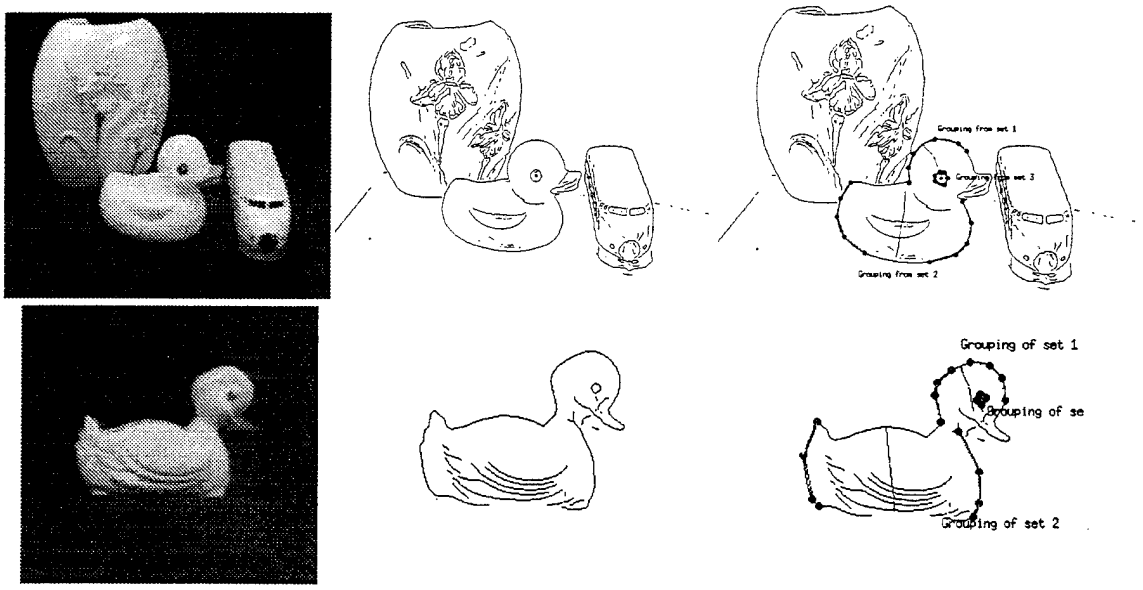
7.5 Results

In Figure 7.7 (top) we show an example of the performance of our current system. The model used for this was one view of an instance of a duck. In the scene however, we had a similar view of another instance of the duck, which was partially occluded. The model gave rise to 21 high level groups resulting in 9 sets out of which 2 were discarded by the reasoning presented in Section 7.3. Among the 7 valid sets contained 6 relevant and 1 was irrelevant. The entire hierarchy took about 13 seconds to compute for the model. The scene gave rise to 47 groups which were organized into 21 sets out of which 5 sets could be discarded. The remaining 16 sets contained 12 relevant sets (out of which 5 were of the duck, and 7 were of the other objects). The sets were used to compute a graph. One of the matched hypothesis is shown. The entire hierarchy took about 2 minutes to compute.

In Figure 7.7 (bottom) we show another example. In this scene the duck was slightly rotated and from a different viewpoint. The scene resulted in 88 curves, which gave rise to 129 groups. These were organized into 19 sets out of which 4 sets could be discarded. The entire hierarchy took about 6 minutes to compute.



Example 1: (left) images of scene and model - duck in scene is different from that in model and partially occluded, (middle) edges of scene and model, (right) example of matched hypotheses.



Example 2: (left) images of scene and model - duck in scene is different from duck in model and partially rotated. (middle) edges of scene and model, (right) example of matched hypotheses.

Figure 7.7 Recognition examples (1) -top and (2) - bottom

7.6 Conclusion

We have developed an approach to use perceptual organization for the purpose of generic object recognition, and show some promising results. Our perceptual grouping is purely data driven. We try to resolve ambiguities and try to discard groups which not necessarily yield any physical interpretation.Our system emphasizes qualitative rather than quantitative tokens and tries to achieve recognition using *spatial correspondences* of these tokens. By using multiple representations for each group, we can deal fairly well with occlusion and scale.By using a set of different views to represent a model we can deal with *incomplete model descriptions*.

Our future work aims at taking care of cases when the system does not find corresponding high level groups (e.g. due to heavy occlusion)? We want to focus on this point by developing a multilevel matching, which allows the system to "fall back" on lower level features in order to find correspondences. We would like to extend the indexing idea directly to the perceptual group sets, rather then by using their approximations. There are several other features and group strategies which we ignore so far: continuation, texture, saliency etc. Including these features would enrich the descriptive and discriminative power of our feature hierarchy.

Our work and the corresponding results in this paper should demonstrate the viability of this approach.

7.7 References

[100]Y. Alavi, G. Chartrand. L. Lesniak, D.R. lick, and C.E.Wall. *Graph Theory with Applications to Algorithms and Computer Science*. John Wiley and Sons, 1985

[101]I. Biederman. Recognition-by-Components: *A Theory of Human Image Understanding*. Psychological Review 1987, Vol. 94, No 2, 115-147.

[102]T . M. Breuel, *Adaptive Model Based Indexing*, In Proceedings of the DARPA IUW, pages 805-814, 1989.

[103]R. A. Brooks. *Model based three dimensional interpretation of two dimensional images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 5(2):140-150, 1983.

[104]J. Canny, *A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, 1986, pages 679-698.

[105]D. Corneil and C. Gotlieb. *An efficient algorithm for graph isomorphism.* Journal of the ACM, 17(1):51-64, January 1970

[106]S. J. Dickinson, A. P. Pentland, and A. Rosenfeld, *3-D shape recovery using distributed aspect matching*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992, pages 174-198.

[107]T.J.Fan. *Describing and Recognizing 3-D Objects Using Surface Properties.* Springer Verlag, New York,1990

[108]P.J.Flynn and A.K.Jain.*3D Object Recognition using invariant feature indexing.* IEEE Workshop on Directions in CAD-Based Vision,115-123,Hawaii, June 1934

[109]W. E. L. Grimson. Object Recognition by Computer - The Role of Geometric Constraints, MIT Press, Cambridge, MA 1990.

[110]T. Kanade. *Recovery of the three-dimensional shape of an object from a single view.* Artificial Intelligence, 17:409-4460, 1981.

[111] D. J. Kriegman and J. Ponce. *On Recognizing and Positioning Curve 3-D Objects from Image Contours.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990, pages 1127-1137

[112]Y. Lamdan, J. T. Schwartz and H.J.Wolfson. *On Recognition of 3-D Objects from 2-D Images.* In Proceedings of IEEE International Conference on Robotics and Automation, April, 1988

[113]D.G. Lowe. *Three-dimensional object recognition from single two-dimensional images*, Artificial Intelligence 31, 1987, 355-395.

[114]R. Mohan and R. Nevatia. *Using Perceptual Organization to Extract 3-D Structures*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No. 11, November 1989, pages 1121-1139.

[115]R.Nevatia and K.Price. *Locating structures in aerial images.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 4(5): 476-484, September 1992.

[116]H. Noltmeier. *Graphentheorie.* de Gruyter, 1976.

[117]B. Parvin and G. Medioni. *A Dynamic System for Object Description and Correspondence*, Proceedings of IEEE CVPR, Jun. 1991, Maui, Hawaii, pages 393-399.

[118]L. G. Roberts. Machine Perception of Three Dimensional Solids. Optical and Electro-Optical Information Processing, pages 159-197, 1968.

[119]H. Rom and G. Medioni. *Hierarchical Decomposition and Axial Shape Description*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Oct 1993, pages 973-981

[120]C. Rothwell, *Hierarchical Object Descriptions Using Invariants.* Applications of Invariants in Computer Vision II, pages 287-302, October 1993, Azores

[121]P. Saint-Marc and G. Medioni. *B-spline contour representation and symmetry detection.* In First European Conference on Computer Vision, pages 604-606, Antibes, France, April 1990.

[122]F. Stein and G. Medioni. *Structural Indexing: Efficient Three Dimensional Object Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 125-145, Feb 1992.

[123]F. Stein and G. Medioni. *Structural Indexing: Efficient Two Dimensional Object Recognition (correspondence)*. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1198-1204, Feb 1992.

[124]F. Stein, G. Medioni, and Parag Havaldar.*Recognizing 3D Objects from 2D Groupings*, Proceedings of the Workshop on Computer Vision in Space Applications, pages 453-464, Antibes, France 1993.

[125]A. Witkin and Tanenbaum. On the role of structure in vision. In J.Beck, B.Hope and A.Rosenfeld editors, *Human and Machine Vision,* pages 481-543. Academic Press, New York, 1983.

[126]S. Ullman and R. Basri. *Recognition by linear combinations of models.* IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 992-1006, Oct 1991

[127]F. Ulupinar and R. Nevatia, *Perception of 3-D surfaces from 2-D contours,* In IEEE Transactions on Pattern Ananlysis and Machince Intelligence, pages 3-18, Jan 1993.

[128]M. Zerroug and R. Nevatia, *Scene Segmentation and Volumetric Descriptions of SHGCs from a Single Intensity Image.* Image Understanding Workshop 1993, pages 905-916.

8 Pose Estimation of Multi-Part Curved Objects

Mourad Zerroug and Ramakant Nevatia

8.1 Introduction

Recognizing 3-D objects from a 2-D image is important for many visual tasks. Part of this problem is the estimation of the 3-D pose of the viewed objects. Alignment, introduced by [134], is a very attractive method since it is simple and efficient. Most objects demonstrated under the alignment technique are those for which low-level image features can be identified and matched with model features. These include polyhedra and objects with sharp corners or distinguished lines. Dealing with complex, curved, objects is more difficult because no such low-level features may be identifiable. This is due to the possible view-dependency of the outlines which may vary wildly with changes in viewpoint and thus are hard to match with object models.

Few efforts have addressed the pose estimation of curved objects. Kriegman and Ponce [136] use a complex method based on elimination theory which finds the pose by minimizing an objective function which is the distance between the viewed silhouettes and the projection of an algebraic surface representation of object models. The method of [131] addresses surfaces of revolution, a somewhat restriced class. A recent method of [139] uses invariants based on the cross-ratio along surfaces of revolution having bi-tangents. The above methods have been demonstrated on relatively simple, though curved, objects.

In this paper, we show that alignment-like techniques can still be used for a large class of complex, curved, multi-part objects provided adequate features and representations are used. More specifically, we demonstrate that high-level descriptions, based on a part-based formalism using generalized cylinders, provide means to establish *quasi-invariant* correspondences (meaning that they are almost exact over almost all viewpoints) between image and model shapes. These correspondences are in terms of powerful intrinsic quantitative shape attributes such as the axis, the scaling function and the cross-section of a part. The idea is that although the outlines may be viewpoint dependent, or may not have distinguished points, the derived shape descriptions in terms of the above powerful attributes (and their combinations) provide viewpoint independent entities which can be put into correspondence with models so represented. We believe this to be an important demonstration of the usefulness of high-level, part-based, descriptions in extending the classes of shapes which can be handled. The classes of shapes demonstrated here currently include arrangements of SHGCs (*straight homogeneous generalized cylinders*), a straight-axis primitive, and

PRGCs (*planar right generalized cylinders*), a curved (planar) axis primitive, a fairly large class of man-made objects (an image of such shapes is shown in Figure 8.1).

Although high-level descriptions have been used in the past in object recognition, they have been used largely for qualitative image-model matching [129,132,140]; we believe their use for quantitative pose estimation to be novel in this work. The method described here is inspired by the results of [146] which demonstrated that GC part-based descriptions can be obtained even in the presence of adverse imaging effects such as clutter and occlusion. Our approach is to use these descriptions to recognize the viewed objects and estimate their 3-D pose.

For lack of space, we will not describe the matching techniques in great detail. Rather, we emphasize the discussion on the use of the high-level part-based descriptions to establish image-model correspondences for relatively complex, curved, structured objects and demonstrate the application of alignment-like methods on those descriptions. However, we would like to emphasize that our method does not assume that model objects with which the viewed objects should be matched are selected by some previous process. Rather, it automatically finds the matching objects from the database and computes their pose.

In this paper, we discuss the computation of the scaled orthographic pose (which is reasonable for each object relatively far from the camera). The results can be used as initial estimates for the computation of the perspective pose as discussed in [134].

We organize the paper as follows. In section 8.2, we describe the representations used and the classes of objects addressed in this paper. In section 8.3, we discuss the use of these descriptions to solve the pose estimation of structured objects, and demonstrate the method on several real images of relatively complex objects. We conclude in section 8.4.

8.2 Representations

In this section, we describe the descriptions used to represent image and model objects.

8.2.1 Image Objects

Image objects are extracted using the method of [146]. This latter produces a graph representation of each viewed object without any prior knowledge of its identity. Rather, it uses purely generic tools to segment objects from the background and decompose their shapes into constituent generalized cylinder parts. A detailed description of the method is given by the authors of [145,146].

An important aspect of the results of that method is that it produces *projective shape descriptions* in that, although 2-D, the image characterizations of parts (such as their image axes and sweeps) correspond to the projection of the 3-D descriptions. This is very useful and, as we will show in section 8.3, allows the establishment of im-

age-model correspondences for objects for which aligment-like techniques haven't been demonstrated yet.

An example of descriptions is shown in Figure 8.1. Each detected object is described as a graph where nodes are parts and arcs labeled joint relationships between parts. For each part, the description consists of the following:

- a label giving its type (SHGC or PRGC)
- a label giving the type of sweep. For an SHGC, this label could be *cylinder, cone* or *non-linear* (the former two indicate a linear sweep while the latter indicates a non-linear one). For a PRGC, ir could be *constant* or *non-constant.*
- the image of the cross-sections if visible. For an SHGC, the "top" cross-section is assumed visible.
- the image of the axis. In the case of SHGCs, for a *cylinder*, this consists of the *direction* of the axis, for a *cone* it consists of the *apex* and for a *non-linear* SHGC of a *line.* For a PRGC, this consists of a list of points which are in practice an excellent approximation of the projection of the 3-D axis [144].
- the pair-wise correspondences between the side boundaries which correspond to the projections of *co-cross-sectional* points in 3-D.
- for SHGCs, the description addtionally includes a list of scaling ratios $\{R_i; i = 1..n\}$ giving the amount of scaling of the cross-section at each position along the axis with respect to the top cross-section. The R_i values are *invariant under scaled orthographic projection* (quasi-invariant under perspective)..

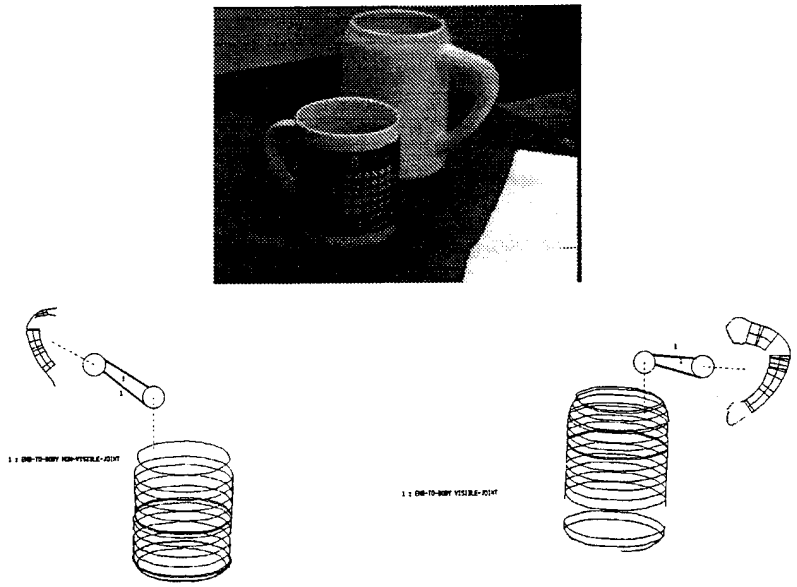


Figure 8.1 Descriptions (2-D) extracted from an image.

8.2.2 Model Objects

Each model object O_M is represented as a graph $G_M = (Q_M, J_M)$ where $Q_M = \{q_M{}^1, .. q_M{}^k\}$ is the set of its GC parts and J_M the set of labeled joints between the parts. Each part is represented by its 3-D intrinsic GC attributes; i.e. the cross-section, the axis and the scaling function all in a 3-D object-centered coordinate system $S_q = (O_q, \vec{i}, \vec{j}, \vec{k})$. Figure 8.2 illustrates this system for SHGCs and PRGCs. More specifically, the attribute representation consists of the following elements:

- the cross-section X_M represented as a list of points; $X_M = \{p_i\}$
- the axis A_M which
  - for an SHGC is represented by the origin O_q (point of intersection of the straight axis with the cross-section plane) and its orientation $\vec{N}_M$
  - for a PRGC is represented by the equation of its plane Π_M and quadratic B-splines giving an analytic expression of the axis points $P_a{}^i$ and their tangent vectors $\vec{T}_a{}^i$
- the scaling function $R_M(z)$ giving for each arclength value z along the axis A_M the ratio R_M of the size of the cross-section at z with respect to the size of a reference (e.g. "top") cross-section

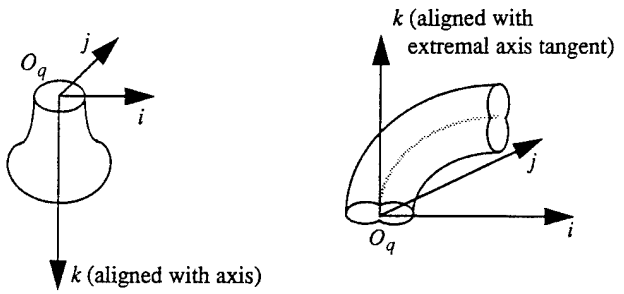


Figure 8.2 Object-centered part coordinate system for an SHGC (a); for a PRGC (b).

Figure 8.3 shows these representations for one of the object models used in the current experimentation.

8.3 Pose Estimation

Given extracted image descriptions, a matching stage is first applied in order to automatically determine which of the model objects O_M corresponds to each image object O_I. This process uses qualitative attributes of both image and model objects (such as part label, sweep type, joint label) in a graph matching method. For lack of space, we will not describe the matching step in this paper. It results in pairings between image and model objects and for each of them, the pairings between image and model

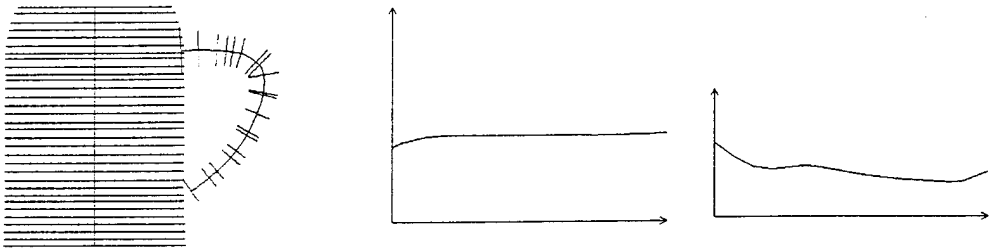


Figure 8.3 Representation of one of the model objects used in the experiments. Left: side view of the object also showing axes. Right: scaling functions of the object's parts (SHGC function is shown first).

parts. Each match (O_I, O_M) can be used to estimate the pose of the viewed object. The method uses the GC-based descriptions to establish correspondences between the image and model parts. We first describe the coordinate systems used and identify the pose parameters.

8.3.1 Coordinate Systems and Pose Parameters

The world coordinate system $W_S = (O, \vec{x}, \vec{y}, \vec{z})$ is chosen so that the x- and y-directions coincide with those of the image plane whose coordinate system is $I_S = (O, \vec{x}, \vec{y})$. For each object, a reference part q_{ref} is chosen. Each part's representation also includes the transformation T_i between its own coordinate system $S_q = (Oq, \vec{i}, \vec{j}, \vec{k})$ and the one of q_{ref}. Currently, we assume that one of the parts is an SHGC which is chosen to be q_{ref} (which of the SHGCs, if many, is not important).

A model object O_M is represented in W_S such that the coordinate system of q_{ref} coincides with W_S. See Figure 8.4.a. Thus, through the transformations T_i and the relationship between the coordinate systems of q_{ref} with W_S, we can determine the pose of each part with respect to W_S. To model the viewing geometry, we use scaled orthographic (weak perspective) projection; i.e. we assume that objects' dimensions are small compared to their distance to the camera. In this model, the depth of a viewed object is taken to be the depth of a reference point $P_0 = (x_0, y_0, z_0)^t{}_{WS}$ on that object. In this case, the projection of a vector $V = P - P_0$ is equivalent to an orthographic projection (along the z-direction) followed by a homogeneous scaling k in the image. Note that scaled orthography is used locally for each object, not the entire scene. As such, each object will have a different scaling factor k. The pose of a viewed object is determined by finding the transformation $\mathbf{T}(P) = \mathbf{R}\,P + \mathbf{t}$ such that the projection of $\mathbf{T}(O_M)$ coincides with O_I; i.e.

$$p = \mathbf{SO}(\mathbf{T}(P)) = \mathbf{SO}(\mathbf{R}\,P + \mathbf{t}) \tag{8.1}$$

where **R** is a rotation matrix, **t** a translation vector, P is any point of O_M (all expressed in W_S), p the projection (belonging to O_I and expressed in I_S) of **T**(P), and **SO** denotes scaled orthographic projection.

Because the third component of **t** is not used, the pose under scaled orthographic projection is determined by 6 parameters, namely 3 for **R** the first 2 components of **t**, say t_x and t_y, and k. We model R as the product of three rotations

$$\mathbf{R} = \mathbf{R}_\phi \, \mathbf{R}_\sigma \, \mathbf{R}_\theta \tag{8.2}$$

where R_ϕ is a rotation about the z-axis by an angle ϕ, R_σ a rotation about the x-axis by an angle σ and R_θ a rotation about the z-axis by an angle θ. The whole sequence of transformations of equation (8.1) is illustrated in Figure 8.4.a-f.

Using the translated origin of the reference model part q_{ref} as the reference point P_0 , the relationship between image and model coordinates is given by

$$u = k\,[\cos\theta\,(\cos\phi\,x - \sin\phi\,y) - \sin\theta\,\cos\sigma\,(\sin\phi\,x + \cos\phi\,y) + \sin\theta\,\sin\sigma\,z] + t_x \tag{8.3}$$

$$v = k\,[\sin\theta\,(\cos\phi\,x - \sin\phi\,y) + \cos\theta\,\cos\sigma\,(\sin\phi\,x + \cos\phi\,y) - \cos\theta\,\sin\sigma\,z] + t_y \tag{8.4}$$

where (u, v) denote image coordinates in I_S and (x, y, z) denote model coordinates in the system of q_{ref} (which, at its original pose, coincides with W_S).

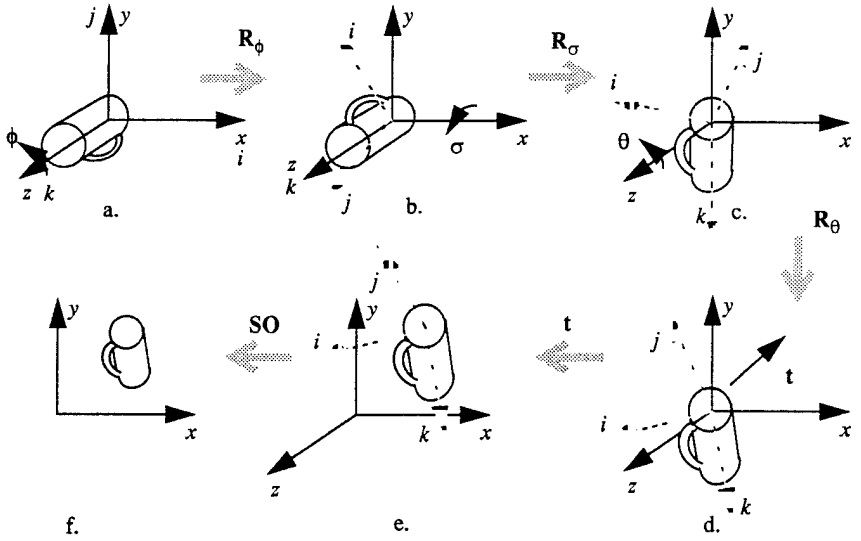


Figure 8.4 Sequence of transformations applied to the model object $\mathrm{O_M}$ so that its image coincides with $\mathrm{O_I}$.

8.3.2 Establishing Correspondences for Single-Part Objects

8.3.2.1 SHGCs

8.3.2.1.1 Non-linear SHGCs

In the case of a non-linear SHGC, for each cross-section X_{Ii} along the surface of the image SHGC ($SHGC_I$), the scaling ratio R_{Ii} (available with the image description) it makes with the "top" cross-section is used to find the position z_i (which we will alternatively refer to as z-value, a missing element in the image description) on the model SHGC ($SHGC_M$) it projects from. This is done by starting from the image "top" cross-section which is matched with the model "top cross-section" whose z-value is z_0=0[1]. Then, moving along the image axis, the z-value z_i corresponding to each image cross-section X_{Ii} along the surface of $SHGC_I$ is chosen to be the one closest (and superior) to the previous z-value (z_{i-1}) such that $R_M(z_i) = R_{Ii}$. This is done by "reading" the model scaling function in a table look-up fashion (see Figure 8.5). This process is not applied to cross-sections lying on constant regions of the scaling function since the correspondences would be ambiguous.

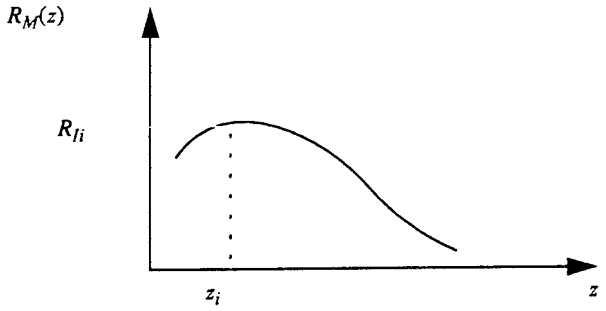


Figure 8.5 Finding correspondences between image cross-sections and model axis positions using the similarity of the invariant scaling ratios.

This results in a set of correspondences,

$$\{(X_{Ii}, z_i)\ i = 0 \ldots n\}$$

between image cross-sections and model positions (equivalently, model cross-sections $X_M(z_i)$) on the SHGC axis. Many image-model point correspondences can now be established. From one of the invariant properties of SHGCs described in [145], the lines joining parallel symmetric points between any pair of cross-sections intersect at a single point (local apex) which belongs to the SHGC axis (Figure 8.6). Thus, the image local apex (A_{Ii}) between each cross-section X_{Ii} ($i > 0$) and X_{I0} corresponds to the model

---

1. the image "top" cross-section may also be matched with the model "bottom" cross-section in case the method fails. This amounts to considering the reverse parameterization of the model SHGC.

local apex ($A_M(z_i)$) between each model cross-section $X_M(z_i)$ ($i > 0$) and $X_M(0)$. The local apexes are determined analytically from the descriptions of the SHGCs.

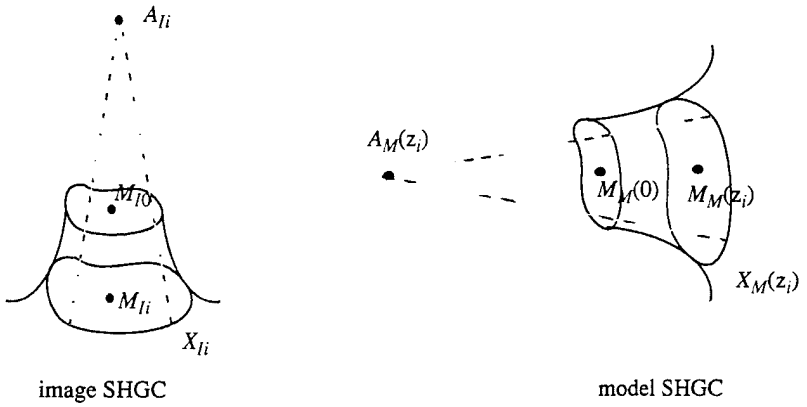


Figure 8.6 Finding axes points correspondences between image and model SHGCs

In the case where the SHGC axis passes through the cross-section mean (centered SHGC) then additional point correspondences can be determined. A correspondence can be established between the mean, M_{Ii}, of each cross-section X_{Ii} and the mean, $M_M(z_i)$, of the corresponding cross-section $X_M(z_i)$. This stems from the fact that, under scaled orthography, the projection of the mean of a closed planar curve is the mean of the projection of the curve. This results in the correspondences

$$\{(M_{Ii}, M_M(z_i))\ i = 0 \ldots n\}$$

Figure 8.7 shows the correspondences $(A_{Ii}, A_M(z_i))$ and $(M_{Ii}, M_M(z_i))$ for the SHGC of the mug in the back of Figure 8.1 (whose origin coincides with the cross-section mean).

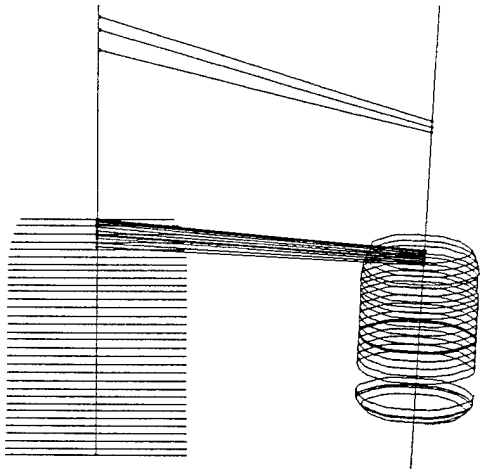


Figure 8.7 Image-model SHGC axis correspondences

8.3.2.1.2 Linear SHGCs

For a conical LSHGC, the local apexes are all a single point (the cone apex). Thus, the set of correspondences between image and model local apexes is a singleton $\{(A_{I0}, A_M(0))\}$. However, the set of correspondences between image and model cross-section means $(M_{Ii}, M_M(z_i))$ includes all visible image cross-sections.

For a cylindrical LSHGC, the local apexes are all at infinity and thus cannot be used. The set of correspondences between image and model cross-section means has two elements, mapping the image and model means of the top and bottom cross-sections of the cylinder. We thus assume that a cylinder does not have a full portion of its surface occluded (it can still be partially occluded such that the full description can be inferred [145]).

8.3.2.2 PRGCs

The image description of an isolated PRGC is not as rich as the one of an isolated SHGC. The main reason is that, unlike with an SHGC, the scaling ratios cannot be determined in the image. In case enough "special" axis points, such as inflections or cross-section corners, exist (and can be identified) then correspondences can be established between image and model PRGCs. But this is not guaranteed. In general, though, it is observed that PRGCs are attached to other parts (serving for example, as handles, etc.). Thus, we choose to discuss PRGCs as part of composite objects.

8.3.3 Establishing Correspondences for Multi-Parts Objects

In some (but practically rare) situations, such as non-centered SHGCs, the sets of points M_{Ii} and A_{Ii} may not be colinear and thus they are sufficient to find the pose of a part (and the whole object if assumed rigid). However, in most cases, a multi-part object typically provides more information about the pose of all its parts, than each one of them does individually. For example, for the mug of Figure 8.1 (back), the presence of the handle conveys the pose of the symmetric cup, whose rotation about its axis is ambiguous when considered alone (the points used for the cup correspondences are all colinear; Figure 8.7).

In this case, we need at least one additional point correspondence which is not colinear with the SHGC axes. Below, we discuss the use of a PRGC attached to an SHGC. The case of two attached (non-colinear) SHGCs is similar. In case, more than two parts exist, any choice of non-colinear axes is equally valid.

Let us denote PRGC_I and PRGC_M the matched image and model PRGCs respectively. In case the axes of PRGC_I and PRGC_M have inflections, then they can simply be used as "distinguished" points to put into correspondence. In case the axes do not have inflections, then we assume that the object's SHGC axis is contained in the PRGC axis plane, as is the case for many objects (due to physical stability reasons).

In the latter case, the tangent line at each point P_j (p_j) of the axis of PRGC_M (PRGC_I) is either parallel to the axis of SHGC_M (SHGC_I) or intersects its supporting

line at some point B_{Mj} (B_{Ij}). Thus, given an image point B_{Ij} (possibly at infinity), if we can determine the corresponding point B_{Mj} then we have a correspondence between p_j and P_j (see Figure 8.8). But, using the previously matched points on the SHGC axes, we can construct corresponding basis vectors V_I and V_M on the axes of SHGC_I and SHGC_M, respectively, with respect to which we can identify corresponding points by the same 1-D coordinate (α) along any of these vectors. Thus, given a point B_{Ij} and its coordinate α_j (i.e. $B_{Ij} = \alpha_j V_I$), B_{Mj} is given by

$$B_{Mj} = \alpha_j V_M. \tag{8.5}$$

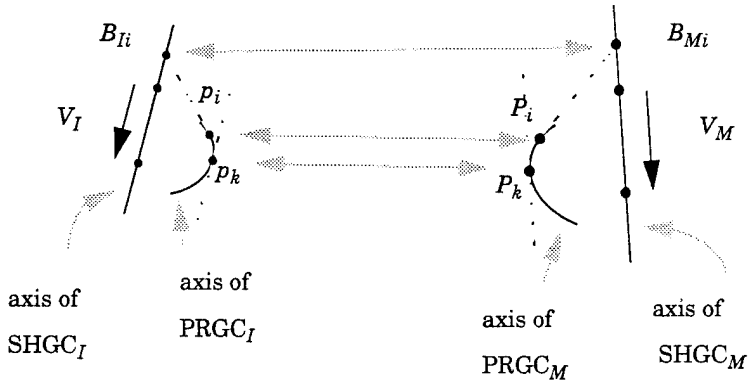


Figure 8.8 Using the SHGC axes correspondences to find PRGC axes correspondences.

In case $\alpha_j \rightarrow \infty$ (i.e. the tangent at p_j is parallel to SHGC_I's axis), then the corresponding point P_j should also have its tangent parallel to SHGC_M's axis. Thus, through the coordinates of points on the SHGC axes, we can now establish an arbitrary number of point correspondences between the axes of PRGC_I and PRGC_M (points whose tangents pass through B_{Ij} and B_{Mj}).

Figure 8.9 shows a correspondence, for the back object in Figure 8.1, between the axes points of PRGC_I and PRGC_M whose tangent lines are parallel to the SHGC axes.

8.3.4 Solving for the Pose Parameters

The image-model point correspondences obtained as described above are not all colinear. A choice of three non-colinear points should be sufficient to determine the 6 pose parameters as discussed by [134]. Here, we briefly review the method.

The angle θ can be determined directly from the image and is given by the orientation of the image SHGC axis. Also, using only the image-model SHGC axes correspondences $(u_j, v_j)^t$, $(0, 0, z_j)^t$, $j = 1, ..m$, we can determine t_x, t_y using a least squares formulation which results in:

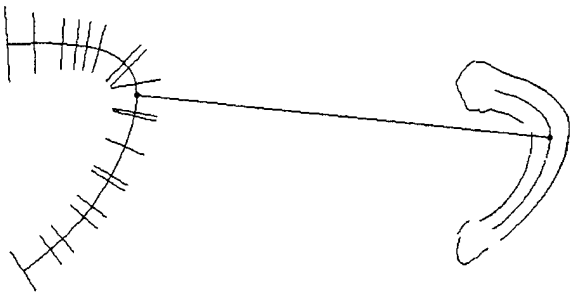


Figure 8.9 An example of image-model PRGC axes correspondence (the corresponding points have tangents parallel to the SHGC axes).

$$t_x = \frac{-\eta \sin\theta \sum_j z_j + \sum_j u_j}{m} \; ; t_y = \frac{\eta \cos\theta \sum_j z_j + \sum_j v_j}{m} \tag{8.6}$$

where

$$\frac{m\sum_j z_j (\sin\theta u_j - \cos\theta v_j) + \sum_j z_j \left(\cos\theta \sum_j v_j - \sin\theta \sum_j u_j \right)}{m\sum_j z_j^2 - \left(\sum_j z_j \right)^2} \tag{8.7}$$

Let $p_a = (u_a, v_a)^t$ and $P_a = (x_a, 0, z_a)^t$ be any pair of matched points not colinear with the above points (for example, the matched points on the image and model PRGC axes, respectively; without loss of generality, the coordinate system of $PRGC_M$ is rotated such that its axis plane contains $SHGC_M$'s i-axis).

Combining equations (8.3) (8.4), and (8.7), it can be shown that we obtain the quadratic form on λ (see [134] for the details of the mathematical formulation):

$$\eta^2 x_a{}^2 \lambda^2 - \left[\eta^2 x_a{}^2 + W_u^{a2} + \left(\eta z_a + W_v^a \right)^2 \right] \lambda + W_u^{a2} = 0 \tag{8.8}$$

where $u_a - t_x = W_u^a$, $v_a - t_y = W_v^a$, and $\lambda = \cos^2\phi$.

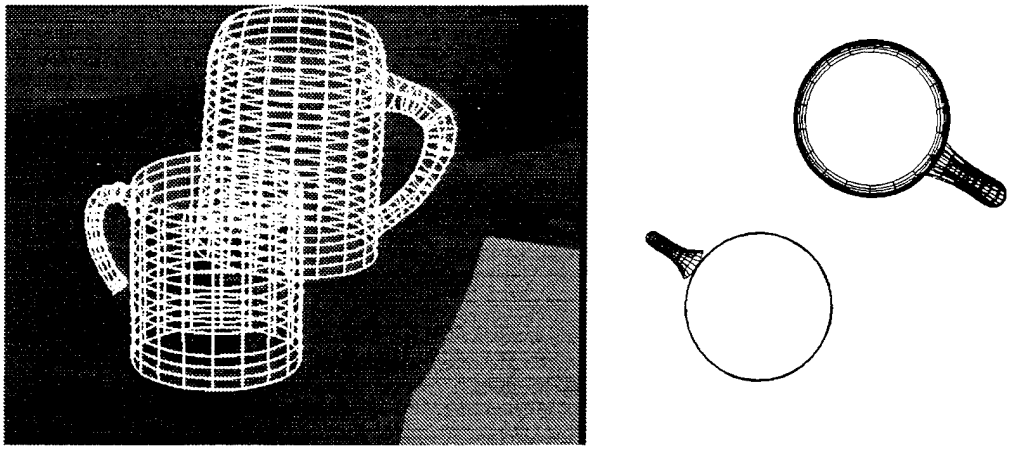
For each solution $\phi = \pm\cos^{-1}(\pm\sqrt{\lambda})$ (if a solution $0 \le \lambda \le 1$ exists), σ and k can be determined by

$$\sigma = \tan^{-1} \left(\frac{\eta x_a \sin\phi}{\eta z_a + W_v^a} \right) + \delta\pi; k = \tag{8.9}$$

where $\delta = 0$ or 1 depending on which results in a positive value of k.

Although the formulation of [134] leaves two possible solutions when three points are used, in our case we can uniquely determine the solution from the rich descriptions used. The solution is selected based on the image labeling of a part's cross-section as facing "towards" ($\sigma > \frac{\pi}{2}$) or "away" ($\sigma < \frac{\pi}{2}$) from the camera (this information is given from the types of junctions which terminate a part). This is another important use of the high-level descriptions in finding the 3-D pose.

The application of this method to the objects of Figure 8.1 is shown in Figure 8.10 showing the transformed and projected model objects (with cross-sections and meridians) on the image. Additional results are shown in Figure 8.11.



a. b.

Figure 8.10 Overlay of the model objects with their estimated poses on the intensity image of Figure 8.1. (a); a top view of the objects (b).

8.4 Conclusion

The pose estimation method exploits the rich GC descriptions in order to establish (viewpoint invariant and quasi-invariant) point correspondences between axes, cross-sections and scaling functions of image and model parts. For the class of shapes addressed here, this results in a closed-form solution and avoids complex methods used in past work on curved objects.

In the near future, we plan to extend the method in several ways. Although the class of objects demonstrated in this paper is more complex than demonstrated in earlier work on curved objects, we plan to extend the method to more general classes, including non-exact primitives. Additionally, we plan to develop an indexing strategy in order to handle large databases. The indexing scheme should also be based on the part-based descriptions of the extracted objects.

Finally, we plan to integrate the current method into a larger framework of "recognition by hierarchical classes". In this framework, the similarity of the image objects is evaluated with respect to known generic classes starting from generic class

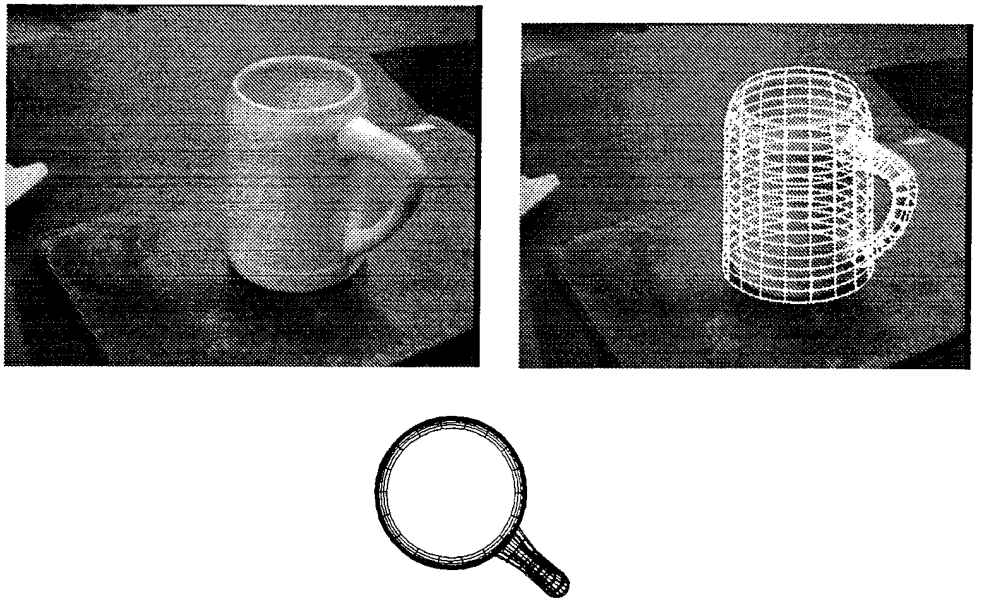


Figure 8.11 Another image of a previous object (back object of Figure 8.1) and its estimated pose.

descriptions and ending at specific object instances. Recognition and pose estimation would then proceed from qualitative criteria, using class descriptions, to quantitative ones, using object instances as shown in this paper. This is important for tasks such as learning and database organization. We expect the representations we use to allow us to make rapid progress on these issues.

References

[129]R. Bergevin and M.D. Levine. Generic Object Recognition: Building Matching and Coarse Descriptions from Line Drawings. *IEEE Transactions PAMI*, 15, pages 19-36, 1993.

[130]R. A. Brooks. Model-Based Three Dimensional Interpretation of Two Dimensional Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):140-150, 1983.

[131]M. Dhome, JT. Lapreste, G. Rives and M. Richetin, "Spatial localization of modelled objects revolution in monocular perspective vision," In Proceedings of *ECCV*, pages 475-485, 1990.

[132]S. Dickinson, 3-D shape Recovery using Distributed Aspect Matching, *IEEE Transactions PAMI*, 14(2):174-198, 1992.

[133]*Geometric Invariance in Computer Vision*, J.L. Mundy and A. Zisserman editors, MIT Press, 1992.

[134]D. P. Huttenlocher and S. Ullman, Recognizing Solid Objecs by Alignment with an Image, IJCV 5, 195-212, 1991.

[135]W.E.L. Grimson, *Object Recognition by Computer-The Role of Geometric Constraints*, MIT Press, Cambridge MA 1990.

[136]D.J. Kriegman and J. Ponce, "On recognizing and positioning curved 3-D objects from image contours," in IEEE Transactions of PAMI, pages 1127-1137, (12) 1990.

[137]J. Liu, J. Mundy, D. Forsyth, A. Zisserman and C. Rothwell, Efficient Recognition of Rotationally Symmetric Surfaces and Straight Homogeneous Generalized Cylinders, In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 123-128, 1993.

[138]D. G. Lowe, "The viewpoint consistency constraint, " IJCV, pages 57-72, 1987.

[139]J.L. Mundy, C. Huang, J. Liu, W. Hoffman, D.A. Forsyth, C.A. Rothwell, A. Zisserman, S. Utcke, and O. Bournez, MORSE: A 3-D object recognition system based on geometric invariants, *IUW*, 1393-1402. 1994/

[140]R. Nevatia and T.O. Binford, Description and Recognition of Complex Curved Objects, *Artificial Intelligence*, 8(1):77-98, 1977.

[141]A. Pentland. Recognition by Parts. in *Proceedings of the ICCV*, pages 612-620, 1987.

[142]L. Roberts. *Machine Perception of Three-Dimensional Solids*. MIT Press, 1965.

[143]F. Ulupinar and R. Nevatia, Perception of 3-D Surfaces from 2-D Contours, *IEEE Transactions PAMI*, pages 3-18, 15, 1993.

[144]M. Zerroug and R. Nevatia, "Quasi-invariant Properties and 3-D Shape Recovery of Non-Straight, Non-Constant Generalized Cylinders", In *Proceedings of CVPR*, pages 96-103. 1993. New York.

[145]M. Zerroug and R. Nevatia, "Segmentation and 3-D Recovery of SHGCs from a Single Intensity Image, in *Proceedings of the ECCV*, pages 319-330, Stockholm, 1994.

[146]M. Zerroug and R. Nevatia, "From an Intensity Image to 3-D Segmented Descriptions" In *Proceedings of the ICPR*, 1994.

9 Representation and Computation of the Spatial Environment for Indoor Navigation

Dongsung Kim and Ramakant Nevatia

We introduce a spatial representation, s-map, for an indoor navigation robot. The s-map represents the locations of obstacles in a planar domain, where obstacles are defined as any objects that can block movement of the robot. In building the s-map, the viewing triangle constraint and the stability constraint are introduced for efficient verification of vertical surfaces. These verified vertical surfaces and 3-D segments of obstacles smaller than a robot, are mapped to the s-map by simply dropping height information. Thus, the s-map is made directly from 3-D segments with simple verification, and represents obstacles in a planar domain so that it becomes a navigable map for the robot without further processing. In addition to efficient map building, the s-map represents the environment more realistically and completely. Furthermore, the s-map converts many navigation problems in 3-D, such as map fusion and path planning, into 2-D ones. We present the analysis of the s-map in terms of complexity and reliability, and discuss its pros and cons. Moreover, we show the results of the s-maps for indoor environments.

9.1 Introduction

Most navigation robots use vision systems to acquire information about the environment in which they navigate. To acquire information, a robot should be able to represent the environment in some way. This depends on various conditions: the tasks to perform, the amount of an *a priori* knowledge of the environment, and the sensors used. A complete and accurate representation is needed as a robot performs sophisticated tasks. For instance, an occupancy map is enough to avoid obstacles, while explicit representation of materials is necessary for landmark recognition.

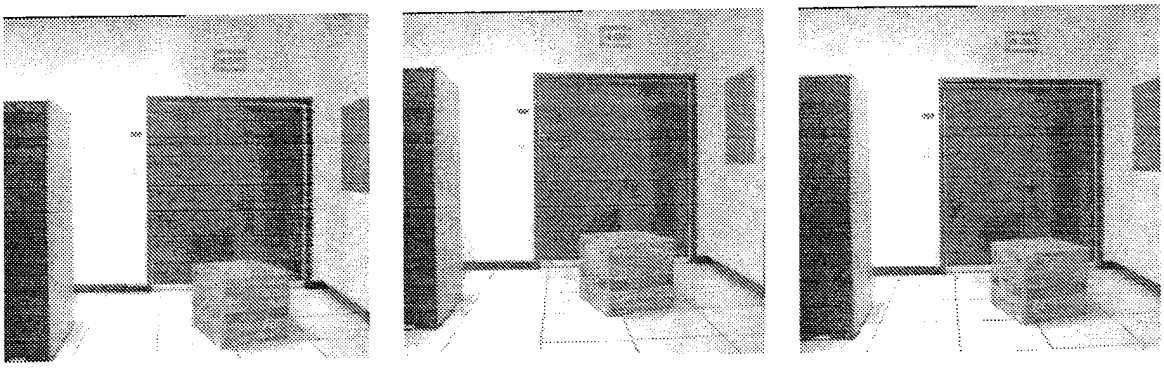
The amount of *a priori* knowledge for the environment determines the completeness of representation of the environment. A fairly complete representation is necessary as a robot has small *a priori* knowledge for the environment. For a robot navigating in a known environment, the environmental representation can be simple low level features, such as edges in 2-D images, because the necessary condition for the robot to reach a goal is to locate itself in the environment. However, when a robot is navigating without a precise map, the environmental representation needs to be more complex. In our case, the robot has general information about the environment, such as flat floors and vertical walls. However, it does not have the specifics of the particular environment; the locations of walls or doors and their widths. We will use the

term "Generic map" to refer such knowledge. For a robot with only a *generic* map, the representation should allow the robot to convert a *generic* map to a specific navigable map.

Furthermore, the sensors used influence the representation of the environment. The main sensors used presently can be divided into two categories: active range sensors and passive range sensors. Using active sensors, an elevation map can be easily acquired because the sensors get dense occupancy information. However, they are expensive and the occupancy information is not enough for robot navigation with only a generic map because object recognition is an important task for such a robot to reach a goal that is described symbolically, for example "turn at the end of a corridor." In the second category, passive stereo can acquire 3-D information of the environment. With the passive stereo, we get only a very sparse depth map. Making a complete map from the sparse depth map is one of the most challenging problems in computer vision.

Little research has been done on the environmental representation for indoor robot navigation with a *generic* map using passive sensors. The current methods for such environmental representation fall into two categories: those methods without surface reconstruction and those methods with surface reconstruction. In the first category, Moravec represented an environment simply with detected features [159], and Braunegg proposed a grid-based presentation using only vertical features [150]. Further, [156,160] tried to represent an environment with a single horizontal slice of a whole 3-D environment. These methods, which do not have surface reconstruction, make incomplete maps. This is because they may miss many surfaces if the surfaces do not have sufficient features or vertical lines. In addition, the method using a single horizontal slice suffers when all horizontal slices of the 3-D space are not the same. For the second category, Bras-Mehlman et al. attempted to make a surface with 3-D Delaunay triangulation and a visibility condition [149,157]. Bruzzone *et al.* first made 2-D Delaunay triangulation then back-projected into 3-D space [151]. Because these methods try to make a surface from adjacent segments, the reconstructed surfaces may be unreasonable, and the constructed free space is shrunk unless there are sufficient segments. Conceptual grouping has also been used to reconstruct surfaces. Mohan and Nevatia attempted to make surfaces by finding rectangles [158]. Chung and Nevatia made surfaces from junctions[154]. Although the reconstructed surfaces are more reasonable, these methods need a lot of computation.

We propose a spatial representation, s-map, for indoor robot navigation using a stereo system when a *generic* map is provided. The environments in which we have experimented are laboratories and corridors in our building, and mostly consist of vertical and horizontal surfaces. A laboratory used in the experiments is shown in Figure 9.1 . Obstacles in the environments have regular shapes that can be linearly approximated. The obstacles used in the experiment are traffic cones, trash cans, boxes, cabinets, desks, etc..



(a)Image 1 (b) Image 2 (c) Image 3

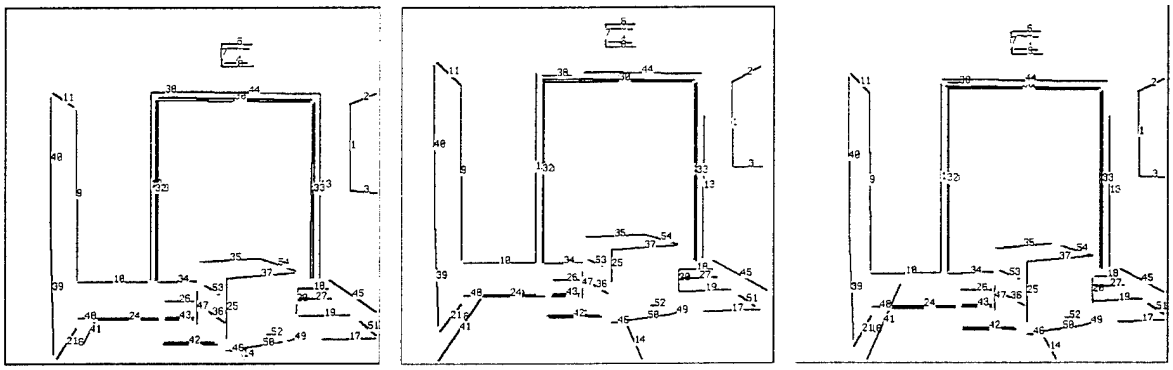
Figure 9.1 Laboratory 1 scene.

We use a Denning mobile robot, Antigone, for our experiments. This robot has ultra sonic sensors that are not used in these experiments, and vision sensors composed of three Sony cameras with Cosmicar lenses of focal length 8mm. The three camera are used to take images. The images are then digitized by a Sun Videopix on a Sun workstation. Then, the edges of the images are detected by a Canny edge detector [152].

We use a junction-supported trinocular stereo system for matching the segments. Our method similar to one developed at INRIA [147,148] but incorporates important junction information [155]. Supporting junctions have been introduced for robust matching, and are defined as the junctions of a segment in one image, which are also maintained in homologous segments in the other images. The supporting junction for a segment is formed by the segment and one representative branch of the segment. The representative branch is selected among many branches to reduce complexity. This junction-supported trinocular matching reduces the ambiguity caused by edge detection error, calibration error, and lens distortion [155]. This stereo system gives segment matches as shown in Figure 9.13 that is the results of stereo matching for Figure 9.9 . Now the matched segments are used to reconstruct 3-D segments for making the s-map.

Making a complete map from a sparse depth map, such as Figure 9.1 , is one of the most difficult problems because of insufficient 3-D information to resolve ambiguities in surface reconstruction. As we see in Figure 9.13 , the nearest segments in either 2-D or 3-D do not form a coplanar surface so that Delaunay triangulation methods make unrealistic surfaces. Moreover, deficient 3-D segments make surfaces shrunk. This problem is severer in conceptual grouping methods. The walls in Figure 9.1 do not have enough segments, so that they can hardly recovered. In addition, the perceptual grouping is not so fast that can be used in robot navigation.

From reconstructed 3-D segments and an image, the s-map represents the locations of obstacles in a planar domain by simple mapping and verification. At first, an



(a)Image 1 (b)Image 2 (c)Image 3

Figure 9.2 Matched segments for Laboratory 1

image is segmented into vertical slices according to the vertical segments. Then segments are distributed among the vertical slices. Now each vertical slice is verified if it can make a vertical surface. The verification is simply done using the viewing triangle constraint and the stability constraint, described in Section 9.2. While the validity of a vertical surface is decided, the segments in its slice are squeezed into an s-map if they can block robot movement..

These simple verification and mapping algorithms allow the robot to build the s-map efficiently. In addition, the s-map can represent the environment more realistically and completely with relatively few 3-D segments because it makes the utmost use of characteristics of indoor environments. Furthermore, the s-map converts many navigation problems in 3-D to 2-D ones because it can represent objects in 2-D domain. This s-map is used for the robot to find a path and navigate the environments.

In Section 9.2 , we introduce an s-map as a spatial Representation For an indoor robot. In Section 9.3, we analyze the algorithm of the s-map. In Section 9.4 , we present the s-map results for indoor environments. Section 9.5 contains the conclusion of this paper.

9.2 S-map

Robot Navigation is defined by the set of tasks that a mobile robot has to perform in order to reach a goal. The tasks involve sensing the environment, analyzing and representing the environment, path planning, and robot locomotion. In this paper, we focus on the environmental representation for the indoor robot using a passive stereo system when only a *generic* map is given. The passive stereo system provides only 3-D segments for the robot to make a navigable map.

The navigable map should represent where obstacles are located, and may represent the shape of the obstacles, when necessary. However, most indoor robots need just the locations of obstacles because they can move only in a plane. Thus, we propose a spatial representation, s-map, which can represent the locations of obstacles in a

planar domain. The term, s-map, comes from the idea that the map is made by squeezing 3-D segments into a 2-D map. In this section, we will first give the definition of the s-map, and exploit characteristics of an indoor environment. Then we will introduce the viewing triangle constraint and the stability constraint, which incorporate the characteristics of an indoor environment to verify vertical surfaces efficiently. We will finally explain how to build the s-map.

9.2.1 Definition of the s-map

The s-map is defined as a map that represents the locations of the visible 3-D surfaces of obstacles in a 2-D space, where 2-D consists of width and length coordinates but does not include a height coordinate. Obstacles are defined as objects that can block the movement of the robot. Therefore, the objects beneath the ceiling are not considered obstacles because they do not block robot movement. Conceptually the s-map is made by first reconstructing a 3-D map and cutting the 3-D map from floor level to robot height level, and finally squeezing the cut 3-D map. However, we do not build the 3-D map, but rather make the s-map directly from 3-D segments.

9.2.2 Characteristics of an indoor environment

The characteristics of an indoor environment are different from those of an outdoor one. Therefore, the characteristics should influence the methodology in order to make the utmost use of them. Now we present the characteristics of indoor environments, and make reasonable assumptions on a navigation environment on which to base the algorithm.

Rooms are mainly composed of vertical walls and horizontal floors, and are filled mostly with man-made objects. Among those man made objects, most objects taller than humans, such as bookshelves, are composed of vertical surfaces. On the other hand, top surfaces of the objects smaller than humans, such as a tea table, can be seen if they are not occluded. From these characteristics, we make three reasonable assumptions for the indoor environment: the floor is flat, and the objects that are taller than a robot, including walls, have vertical surfaces. Moreover, top surfaces of the smaller objects than the robot can be detected at least partially when they are visible.

In order to use these characteristics, we divide objects into two categories: those that are smaller than the robot and those that are taller. Locations of the smaller objects can be represented by simply dropping height information of the 3-D segments of the objects because the robot can see all the boundaries of visible surfaces. Therefore, the locations of the smaller objects are acquired by squeezing the 3-D line segments of the smaller objects to the floor. For the taller objects, the robot may not see all the boundaries of visible surfaces. Horizontal boundary lines generated by an object and either the floor or the ceiling may not be seen due to occlusion by other objects or the small viewing angle of a lens. However, the vertical boundary lines are likely to be seen in most cases because they are tall enough not to be occluded when they are within viewing angles. These vertical lines are important clues to find the loca-

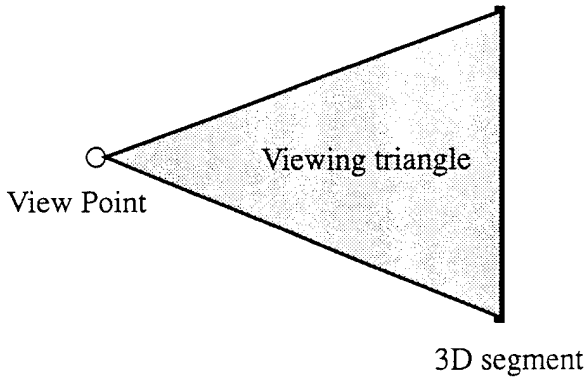


Figure 9.3 Viewing Triangle

tions of the taller objects including walls. If we follow the same process as we do for the smaller objects, we get only points because the vertical lines are points in the 2-D space, which consists of width and length but not of height. Therefore, we need some kind of surface reconstruction with the vertical lines.

The surface reconstruction with vertical lines uses the characteristic of a vertical surface, that is, the vertical surface has two adjacent vertical lines bounding it. However, all adjacent vertical lines are not in the same surface. Two adjacent vertical lines fall into two categories: both of them are on the same surface or they are on different surfaces. Those two adjacent vertical lines that are on the same surface can make a surface. However, those two adjacent vertical lines that are on the different surfaces cannot make a surface. Therefore, vertical surfaces are reconstructed only from the two adjacent vertical lines on the same surface. Then the reconstructed vertical surfaces are squeezed to a floor. Now the problem of surface reconstruction with the vertical lines becomes how to verify if two adjacent lines are on the same surface. The verification process is described in section 9.2.3 .

9.2.3 Viewing triangle constraint

We assume that objects are opaque. Seeing an object means that the visible part of the object is not occluded by any other obstacles. Similarly, a 3-D segment corresponding to the edge detected in an image, is not occluded by any other objects. From the above observation, we introduce the viewing triangle constraint described below. Before explaining the viewing triangle constraint, we define the viewing triangle as follows.

Definition: Viewing triangle *The triangle made with a view point and the 3-D segment corresponding to the edge detected in an image*

Figure 9.3 shows a viewing triangle..

Now the viewing triangle constraint is stated as follows.

Observation 1 : Viewing Triangle Constraint *The space confined within a viewing triangle is free space.*

This viewing triangle constraint is used to verify whether two adjacent vertical lines are on the same surface and make a vertical surface. Before explaining how to use the viewing triangle constraint for verification, we describe how to find adjacent vertical lines. First, a vertical line is found in a 2-D image, and the verticality of the vertical line is checked with its 3-D information. Moreover, The vertical line should be also taller than a robot because they are assumed to be boundary vertical segments of vertical surfaces that are taller than a robot. Then, for the vertical line, an adjacent vertical line, which also satisfies the above conditions, is searched for in the image.

After finding two adjacent vertical lines, we collect segments between them. Collecting such segments is done by simply checking column coordinates of segments. Because a vertical segment has the same row coordinate value along the column coordinate, its column coordinate can represent its location in the image. Therefore, segments are collected into a vertical slice made by the two adjacent vertical lines if column coordinates of the segments are located between column coordinates of the two adjacent vertical segments. The segments are then categorized into three groups: coplanar, inhibition, and irrelevant segments.

- coplanar segments : The segments that are coplanar with the hypothesized vertical surface made with two adjacent vertical lines.
- inhibition segments : The segments that are either behind or across the hypothesized vertical surface
- irrelevant segments : The segments that are in front of the hypothesized vertical surface.

When the presence of the surface is verified, the verification is done conservatively to prevent the robot from hitting obstacles. When the robot misses a real wall, it collides with it. Making an imaginary wall only reduces free space, and the reduced free space can be expanded by the following image sequence. Therefore, we presume that there is a wall unless there is evidence of nonexistence of the wall. The evidence comes from the segments in its vertical slice.

The viewing triangle constraint generated by coplanar segments helps two adjacent vertical lines make a vertical surface. However, the viewing triangle constraint generated by inhibition segments prevents the two adjacent vertical lines from making a vertical surface. As the name suggests, irrelevant segments do not affect the making of a vertical surface with the two adjacent vertical lines. Thus verification of presence of a vertical surface for two adjacent vertical lines is done by checking if there are inhibition segments between the two adjacent vertical lines. This verification using inhibition segments leads the following observation.

Observation 2. Verification with viewing triangle constraint *Two adjacent vertical lines can make a vertical surface if the surface does not violate the viewing*

triangle constraint generated from the segments between the two adjacent vertical lines.

9.2.4 Stability constraint

We introduce perceptual grouping to extend surfaces that are missed by the reconstruction using the viewing triangle constraints. Although much research has been done on grouping for surface reconstruction both in 2-D and 3-D, the research has tried to group surfaces by hypothesize-and-test methods with geometric properties. One drawback of this approach is that it requires a lot of computation. Therefore, it is not practical for robot navigation where an important concern is real time navigation. We try to achieve speed by searching small areas with the help of the stability constraint. Before explaining grouping, we describe the stability constraint. Then we describe the cases when perceptual grouping is necessary, and the perceptual grouping to extend surfaces.

The stability constraint is described as follows.

Stability constraint *Every object must be stable with respect to gravity.*

Pin-like objects, such as vertical segments, tend to fall down unless there are some supports. Conversely, if there is a vertical segment, then there are supports for it. The supports can be any segments adjacent to the vertical segment, which can keep the vertical segment vertical. In an indoor environment, the vertical segments are assumed to be side ends of vertical surfaces. Thus the supports are the segments that are adjacent to the end points of the vertical segments and are either on a stable plane or beneath a stable plane. Figure 9.4 illustrates the supports for a vertical segment.

Grouping segments of the same vertical surface is necessary when one of our assumptions is loosely preserved. A basic assumption of reconstructing a vertical surface is that both vertical end lines for the surface are visible, but that assumption is not always true. The missing end line occurs in cases of clipping or occlusion. The clipping case occurs when a wall is so large that entire wall is not covered in an image. In the second case, a wall is occluded by other vertical objects, such as a cabinet, so that vertical lines of other objects locate between the two vertical lines of the wall. These two cases cause missing vertical surfaces.

The missing surfaces can be extended if there is a support. When there is more than one support, the farthest support line is selected among them. The support and the vertical line can make a vertical surface if the surface hypothesized by them does not violate the viewing triangle constraint. For easy implementation of the verification, a ghost vertical line is introduced. The ghost vertical line for a vertical line is an imaginary vertical line that is located at the end point of a support line of the vertical line. The ghost vertical line for the clipping case is made when vertical segments are collected, while the ghost vertical line for the occlusion case is made when a vertical slice has inhibition segments. After finding a ghost vertical line, the vertical slice

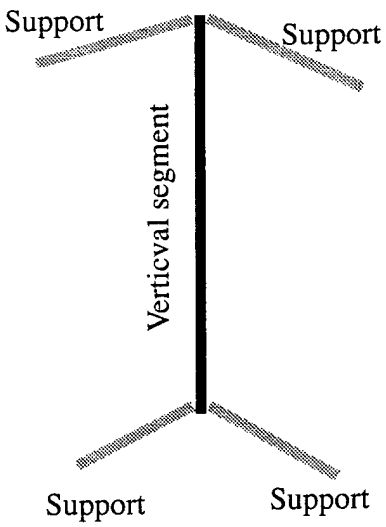


Figure 9.4 The supports for a vertical segments

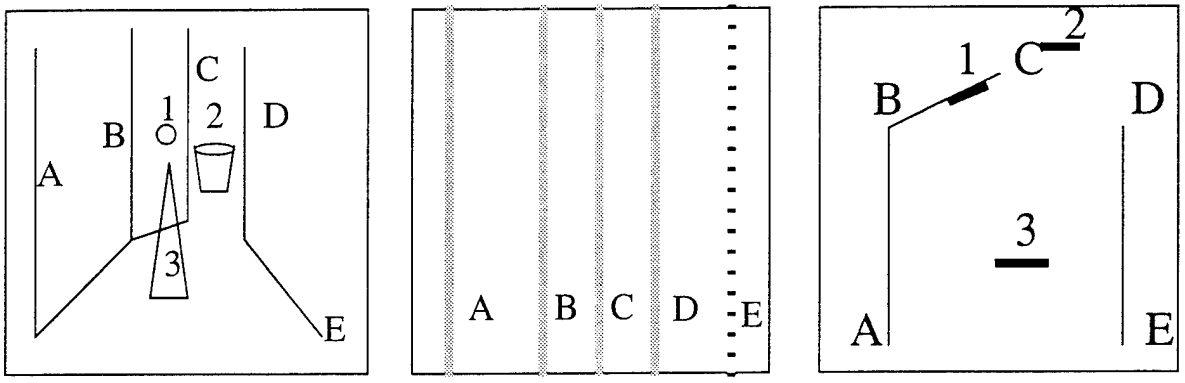
made by a vertical line and its ghost vertical line does the same verification as ordinary vertical slices do.

9.2.5 Algorithm for making the s-map

The algorithm for making the s-map is in two steps: segmentation of an image into vertical slices, and verification of vertical surfaces for the vertical slices while mapping smaller objects into the s-map. The first step is further divided into three modules: collecting vertical segments taller than a robot, dividing the image among vertical slices according to the collected vertical segments, and distributing other segments among the vertical slices. The algorithm for making a s-map is summarized in Figure 9.5 . The details of the algorithm are given below.

In the first step, we first collect vertical segments including ghost vertical lines. Next, we divide an image into vertical slices according to the vertical segments. Finally, the remaining segments, such as nonvertical segments and the vertical segments smaller than a robot, are distributed among the vertical slices. While being distributed, the segments are categorized into three groups: inhibition segments, coplanar segments, and irrelevant segments. Each segment is distributed among those vertical slices that lie in between the begin and end points of the segment.

At the second step of making an s-map, we verify the presence of the vertical surface hypothesized by a vertical slice while mapping possible obstacles into the s-map. If there are no inhibition segments, a vertical surface is presumed existing. Otherwise, a support for a vertical segment is searched for. In the case, where no support for the vertical segment is found, no further attempt is made to find a partial vertical surface. In the other case, where a support for a vertical segment is found, a partial



a) Edges b) Vertical lines c) S-map

Figure 9.6 Examples of making an s-map

vertical surface is hypothesized with the support and the vertical segment. The hypothesized vertical surface is verified using the viewing triangle constraint, that is, if there is no inhibition segment for the surface hypothesized, the surface is presumed existing.

While verifying the presence of a vertical surface, the segments smaller than the robot are mapped to an s-map by simply dropping height information because the robot can see the boundaries of the top surfaces as described in section 9.2.2.

One example of making an s-map is shown in Figure 9.6 -a, b, and c. One edge image for an indoor scene is given in Figure 9.6 -a. In this figure, we assume that A, B, C, and D are vertical lines and that segments group 1, 2, and 3 are coplanar segments, inhibition segments, and irrelevant segments, respectively. First, this image is segmented. From this image, we can make vertical slices according to vertical lines A, B, C, D, and E. E is a ghost vertical line. Figure 9.6 -b shows the vertical lines. Now other segments are distributed among vertical slices. Segment groups 1 and 3 are distributed into the slice made with B and C, and segment group 2 is distributed into the slice made with C and D. While distributing the segments, we classify the segments into one of three groups: coplanar, inhibition, and irrelevant groups. At the second step, verification is done. The slice made from A and B reconstructs a vertical surface because there are no inhibition segments, and then the vertical surface is mapped to an s-map. Similarly, the slice made from B and C is mapped to the s-map while segments group 3 is mapped to the s-map because the segment group 3 can be an obstacle for the robot. However, the slice made from C and D cannot form a vertical surface because there are inhibition segments that constitute segment group 2. Instead, the segment group 2 is mapped to the s-map because it can block the movement of a robot. The slice made from D and E is mapped to the s-map because there are no inhibition segments. The s-map of Figure 9.6 -a is shown in Figure 9.6 -c.

1) Segmentation of an image

a) We collect vertical lines taller than the robot.

b) The image is segmented into vertical slices according to the vertical lines.

c) Other segments are distributed among the vertical slices. The segments in a slice are categorized into the three groups: coplanar, inhibition, and irrelevant groups.

2) Verification and mapping

a) In each slice, the validity of the hypothesized vertical surface is checked in terms of the viewing triangle constraint.

If the vertical surface is valid one

then the surface is squeezed into the s-map.

else a support for a vertical segment is searched for.

If there is a support
and the surface made by the support
and the vertical is a valid surface
then the surface is squeezed into
the s-map.

b) While the validity of a vertical surface is decided, the segments in its slice are squeezed into an s-map if they can block robot movement.

Figure 9.5 Algorithm for building a s-map

9.3 Discussion of the s-map

9.3.1 Complexity of the s-map

The s-map algorithm is analyzed for its complexity. Let's assume that there are n matched segments in an image. Selection of vertical lines takes $O(n)$, and dividing the image needs sorting of the vertical lines. Let the number of vertical lines l and the number of columns c. Because vertical lines have the same column coordinate, the vertical lines can be sorted by a bucket sort method where buckets are image columns. Thus the complexity of sorting is $O(max(l,c))$. Now nonvertical segments are distributed among slices. They are distributed to those slices that the nonvertical segments straddle. These slices are simply the slices lying between the column coordinates of their end points. The worst case is that all nonvertical segments straddle all vertical segments, where the complexity is $O(l \times (n\text{-}l))$. The maximum of this complexity occurs when l is $n/2$. Therefore, the worst case complexity is $O(n^2)$. In real cases, most nonvertical segments straddle less than three vertical slices and the number of vertical slices is less than ten because the number of walls and tall objects is small in indoor

environments. Thus we estimate the complexity to be $O(n)$ in real cases though a much deeper analysis is necessary. Finally, the complexity of the verification algorithm is the same as that of distribution. The complexity of s-map is $O(n^2)$ in worst case, and $O(n)$ in real cases.

The run time of the making the s-map from reconstructed 3-D segments was about one tenth of a second in our experiments. The algorithm was run in the Sun Sparcstation 10. The algorithm is currently programmed in Lisp. Moreover, the current program is not an optimized one. Thus the run time can be reduced by optimizing it. In addition, the run time can be further reduced by mutiprocessors because each slice can be verified independent of the other slices.

9.3.2 Reliability of the s-map

Graceful degradation is one of the important aspects for vision algorithms because vision algorithms deal with real pictures whose quality varies according to environmental changes. For the s-map algorithm, there are two places where errors come in: Edge detection and matching. In edge detection, the most common errors are missing edges, shortened or broken edges, and mislocated edges. These errors affect the performance of not only matching but also the s-map algorithm. In matching, common errors are wrong matches, missing matches, and shortened matches. From these two levels of errors, the s-map algorithm suffers from four kinds of errors: partial matches, missing matches, mislocated matches, and wrong matches. For these four cases, the reliability of the algorithm is explored.

In the first case, partial matches do not affect taller object surfaces, but shrink smaller object surfaces when the partial matches are segments of top surfaces. In reconstructing a vertical surface, the necessary information constitutes locations of vertical lines and coplanar segments, as described in section 9.2.4. Therefore, partial matches do not degrade the algorithm of reconstructing vertical surfaces. In reconstructing the smaller objects, only segments of top surfaces are necessary. Thus only partial matches of top surfaces shrink smaller objects. The shrunk surfaces may be extended by grouping the top surfaces. However, such extensions are not necessary in most navigation problems because the shrunk portion of an object is relatively small and does not cause problems for a robot to plan a path with the shrunk object.

In the second case, missing matches can degrade the s-map algorithm more severely. For simple analysis, we assume that every surface is a quadrilateral. For taller objects, even two missing matches are tolerable if they are nonvertical lines. In the case of missing either of the two vertical lines, the shrunk vertical surface can be reconstructed if there is a support. For smaller objects, missing those matches that are not top surface boundaries does not degrade the algorithm of the s-map at all because top boundaries represent the location of the objects in the s-map. Missing some of top boundaries means that an object is shrunk. Even in the case where only one of the top boundaries is detected, the location of the object can be still known.

Finally, both mislocated matches and wrong matches show the same error pattern because both of them generate a 3-D segment having wrong 3-D information. They cause more damage to taller objects if they are vertical lines of the taller surfaces. They shift location of the vertical surfaces. However, for smaller objects, they do not affect the location of the object. They just place themselves in wrong locations because the s-map does not make surfaces by combining boundary lines of smaller objects, and may prevent two adjacent vertical segments from making a vertical surface when the wrong matches become inhibition segments of them.

9.3.3 Advantages and disadvantages of the s-map

The advantages of the s-map are fourfold: Making the map is very simple. The map is directly made from 3-D segments, so it is not necessary to reconstruct a 3-D open space. The second advantage is that the s-map itself is a navigable map. Therefore, we do not have to cut a 3-D space to find a navigable map. In conventional methods [157,151], it is very difficult to find a cutting height level of a 3-D space for navigation. Since a reconstructed 3-D space is produced by visible line segments and the lower part of objects may be invisible, the reconstructed 3-D space could be too conservative at the lower level of a navigation environment so that only small free area is available for navigation. Thus, finding a cutting level is a compromise between area of navigable space and safety. Therefore, deciding the cutting level is difficult. Thirdly, the s-map represents the environment more realistically and completely with relatively few 3-D segments. This comes from the utmost use of characteristics of indoor environments, which are embedded in the viewing triangle constraint and the stability constraint. Finally, the s-map can turn many 3-D problems in navigation into 2-D problems, for example, path planning and map fusion. Therefore, many navigation problems can be solved efficiently. The disadvantage of the s-map is that wrong matches may prevent two adjacent vertical segments from making a vertical surface when the wrong matches become inhibition segments.

9.4 Results

We have tested the s-map in indoor environments, such as corridors and laboratories. Moreover, our robot has successfully navigated corridors in our building by representing the corridors with the s-map [155].

In this paper, we present the s-maps for two laboratory image frames and one corridor image frame, which are illustrated in Figure 9.8 , Figure 9.11 , and Figure 9.14 . As we can see in these s-maps, the accuracy of the s-maps depends on the accuracy of the reconstructed 3-D segments. Therefore, a more accurate s-map can be acquired with more accurate the reconstructed 3-D segments. This depends on accuracy of a stereo system.

Figure 9.1 shows an image frame for a laboratory. Moreover, Figure 9.1 displays matched segments of the Figure 9.1 . Now vertical segments are selected. Figure 9.7 illustrates the selected vertical segments. Among the vertical segments,

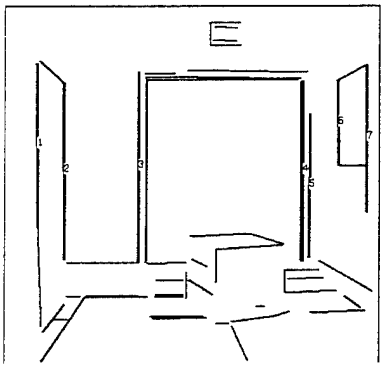


Figure 9.7 Vertical segments of Laboratory 1

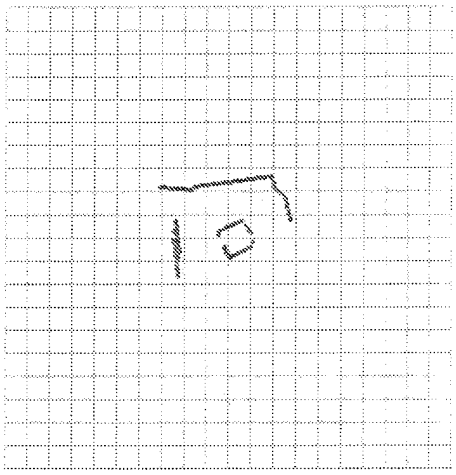
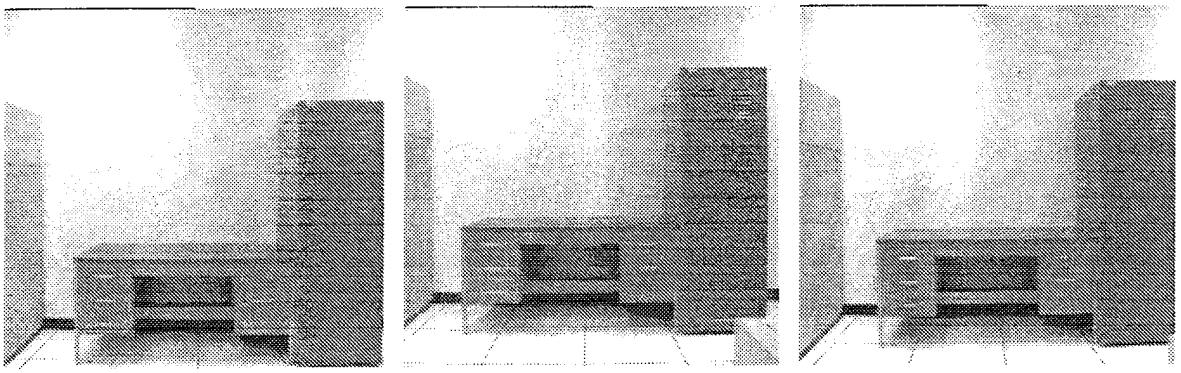


Figure 9.8 S-map of Laboratory 1

the vertical segment 7 is a ghost vertical segment made by the support of the vertical segment 6. Then the verification and the mapping are done. The locations of two walls, a box, and a cabinet are successfully represented in the s-map. Moreover, the right wall is extended with the vertical and its support: Segment 1 and Segment 2 in Figure 9.1 . Similarly, the left wall that is between the cabinet and the door, is also extended with the vertical segment and its support: Segment 12 and Segment 10 in Figure 9.1 .

Figure 9.9 shows an image frame for a laboratory. Moreover, Figure 9.10 displays matched segments of Figure 9.9 . The s-map is given in Figure 9.11 . The locations of a desk, an air conditioner, and a cabinet are represented in the s-map. A wall behind the desk is also represented in the s-map. Moreover, front part of the cabinet is extended with the vertical segment and its support: Segment 37 and Segment 35 in Figure 9.10 . Similarly, the air conditioner is also extended with the vertical segment and its support: Segment 23 and Segment 22 in Figure 9.10 .



a) Image 1 b) Image 2 c) Image 3

Figure 9.9 Laboratory 2

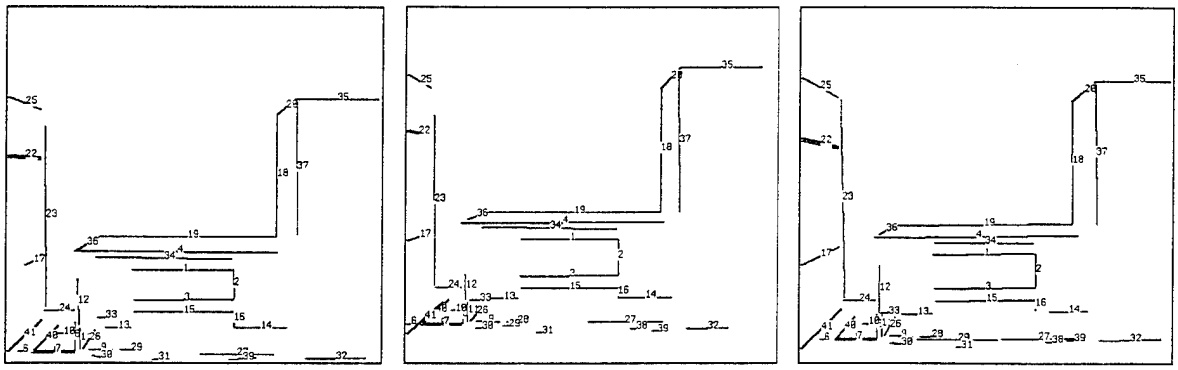
Figure 9.12 shows an image frame for a corridor. Moreover, Figure 9.13 displays matched segments with the s-map given in Figure 9.14 . Two walls of the corridor and one wall at the end of the corridor are represented. Moreover, two traffic cones are successfully represented.

9.5 Conclusion

In this paper, we introduced a new representation of an indoor environment, s-map, which can represent the location of obstacles in a planar map. The obstacles are defined as any objects that can block robot movement. The s-map utilizes the viewing triangle constraint and the stability constraint. These constraints provide both efficient validation of the vertical surface hypothesized by two adjacent vertical lines and efficient grouping of the partial vertical surface having a single vertical line. The s-map is made by mapping the 3-D segments of smaller objects and verified vertical surfaces into a planar map. The mapping is done by dropping height information of the 3-D segments or the vertical surfaces. Therefore, this mapping allows the s-map to be made directly from 3-D segments without reconstructing a 3-D open space and to represent obstacles in a planar map that is a navigable map for the robot moving in a plane. In addition to the efficient map making, the s-map transforms many 3-D problems in navigation into 2-D problems because it represents objects in 2-D domain. However, the s-map is applicable only in limited environments because it assumes flat horizontal floors and vertical walls, and it degrades when there are wrong matches becoming inhibition segments. Our future effort will be directed toward extending the s-map for more complex environments and adding robustness in validating vertical surfaces against wrong matches becoming inhibition segments.

9.6 References for Chapter

[147]N. Ayache. *Artificial Vision for Mobile Robots*. The MIT Press, 1991.



a) Image 1 b) Image 2 c) Image 3

Figure 9.10 Matched segments for Laboratory 2

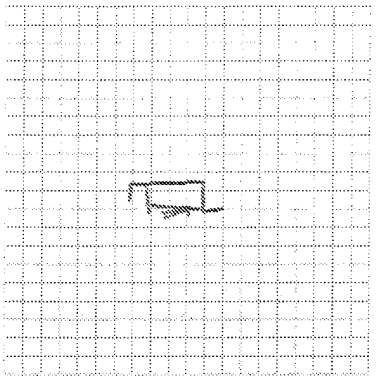
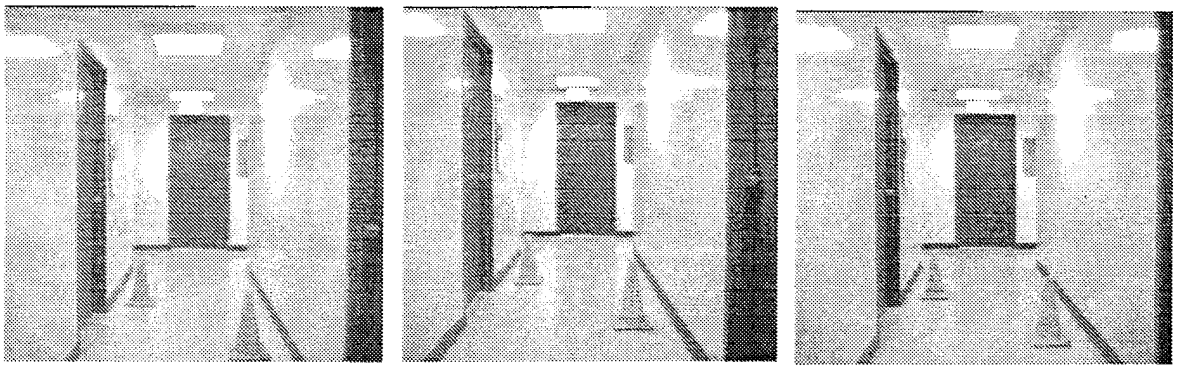
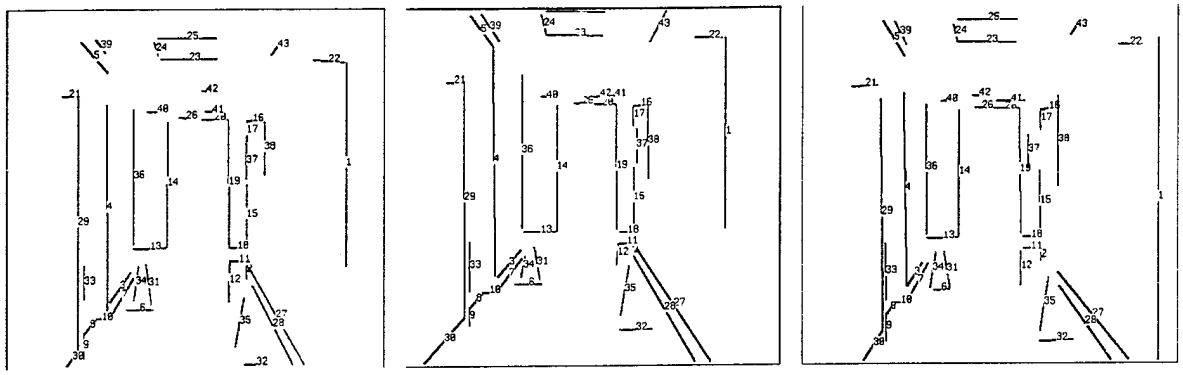


Figure 9.11 S-map for laboratory 2



a) Image 1 b) Image 2 c) Image 3

Figure 9.12 Corridor Scene



a) Image 1 b) Image 2 c) Image 3

Figure 9.13 Matched segments for Corridor

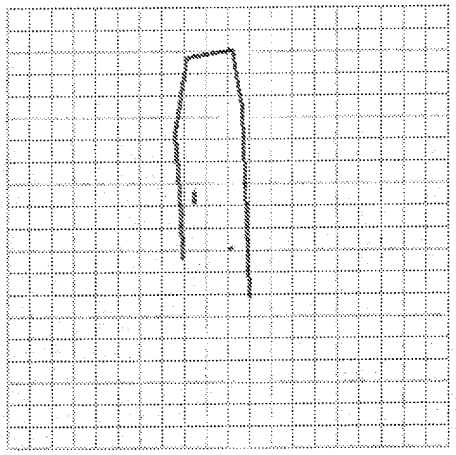


Figure 9.14 S-map for Corridor

[148]N. Ayache and F. Lustman. Fast and reliable passive trinocular stereovision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 422–427, London, England, June 1987.

[149]J.D. Boissonnat, O.D. Faugeras, and E. L. Brass-Mehlman. Representing stereo data with the delaunay triangulation. *IEEE Int. Conf. on Robotics and Automation*, 1988.

[150]D. Braunegg. Marvel: A system that recognizes world locations with stereo. *Procs. of Int. Conf. on Computer Vision and Pattern Recognition*, 1991.

[151]E. Bruzzone, M. Cazzanti, L. De Floriani, and F. Mangili. Applying two-dimensional delaunay triangulation to stereo data interpolation. *European Conference on Computer Vision*, 1992.

[152]J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[153]Y. Chen and G. Medioni, "Object Modelling by Registration of Multiple Range Images," *International Journal of Image and Vision Computing (IVC)*, 10(3):145–155, April 1992.

[154]C.-K. R. Chung and R. Nevatia. Recovering building structures from stereo. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, Palm Springs, CA, December 1992.

[155]D. Kim and R. Nevatia. Indoor navigation without a specific map. *Intelligent Autonomous Systems*, 1993.

[156]D. J. Kriegman, E. Triendl, and T. O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Trans. on Robotics and Automation*, 5(6), December 1989.

[157]E. Le Bras-Mehlman, M. Schmitt, O. D. Faugeras, and J. D. Boissonnant. How the delaunay triangulation can be used for representing stereo data. *Procs. of Int. Conf. on Computer Vision*, 1988.

[158]R. Mohan and R. Nevatia. Using Perceptual Organization to Extract 3-D Structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1121–1139, November 1989.

[159]H. P. Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.* PhD thesis, Stanford University, Stanford, California, September 1980. Technical Report AIM-340 and STAN-CS-80-813.

[160]M. H. Soldo. Reactive and preplanned control in a mobile robot. *Procs. of Image Understanding Workshop*, 1990.

10 A Method for Recognition and Localization of Generic Objects for Indoor Navigation

Dongsung Kim and Ramakant Nevatia

We introduce an efficient method for recognition and localization of generic objects for robot navigation, which works on real scenes. The generic objects used in our experiments are desks and doors as they are suitable landmarks for navigation. The recognition method uses significant surfaces and accompanying functional evidence for recognition of such objects. Currently, our system works with planar surfaces only and assumes that the objects are in a "standard" pose. The localization and orientation of an object are represented with the most significant surface in an "s-map." Some results for laboratory scenes are given.

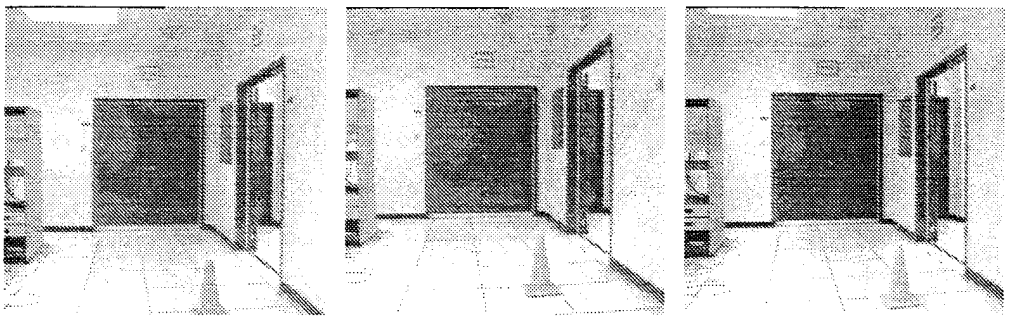
10.1 Introduction

Our goal is to provide visual capabilities for a robot to navigate in indoor environments such as an office building. For this, not only must the robot be able to sense the objects in its environment for the purpose of obstacle detection but also recognize some of them to be used as landmarks for navigation. One approach to this task could be to provide a detailed map of the objects and structures in the environment to the robot. This allows conventional model-based object recognition techniques to be used for landmark detection and path planning. This strategy, however, has several limitations. First, the objects and their arrangement in an indoor setting are constantly changing. Even normally stationary objects, such as furniture, may be moved occasionally. Also, providing detailed geometric models for all objects even in a single room can be a very difficult and tedious task. When a common object, such as a desk, is replaced by another one, completely new models may have to be provided even if the two objects serve similar functions.

To overcome these difficulties, we propose to represent the objects and structures by some *generic* models. This enables the objects to be recognized as belonging to a certain class without having to also determine which specific one. Such generic modeling allows the robot to navigate in environments without knowledge of the specific instances or their locations.

An example of a scene that the robot must handle is shown in Figure 10.1. The robot may be asked to use a desk as a landmark and to pass through a door after the desk. We only wish to provide generic descriptions of the room and the objects to the robot. A room can be thus characterized by horizontal doors and vertical walls that

may have doors. The room may consist of objects, such as desk, whose generic properties are known to the robot, but also other unknown objects which can not be used as landmarks but nonetheless must be avoided during navigation. In this paper, we will focus only on the recognition of the generic classes.



(a) Image 1 (b) Image 2 (c) Image 3

Figure 10.1 Door 1

Little research has been done on recognition of generic classes of objects in computer vision. It is somewhat difficult to precisely define the notion of a generic class, but it surely excludes precise geometric models. The most generic representation of an object is probably in terms of its functionality. Thus, concept of a door is that of an opening allowing passage of objects and a means of closing this opening. The concept of a desk is that of an object that allows a human to place objects on it and work on them in a comfortable posture. The inference of such high level functionality from real images is, however, quite difficult. Some attempts have been made towards this [Star-Bowy91], but these systems do not take images as inputs. In early work, Tenenbaum and Garvey [Garv-Tene74,Tene73] attempted to recognize objects in an office scene by using point properties; however, such properties are not sufficient to distinguish among complex objects in a complex scene, in general. Another approach is to view various instances of a class as generated by varying parameters of a parametric representation. Parametric representations have been studied by several researchers [Broo83,Grim89,Lowe87,Marr-Nish77,Neva-Binf77], however, such approaches do not naturally capture the common variations found in everyday objects such as desks.

In this paper, we propose an approach of representing objects by their significant surfaces and by relations among them, which we believe is sufficiently general for recognizing common objects in an indoor environment and can work with real scenes. Significant surfaces are chosen based on their functional role. A desk is thus characterized by a working surface and some surfaces that correspond to the support structures, that is "legs." The working surface is characterized by some properties, such as a range of sizes and heights. Besides the observation of surface properties, the surfaces can also be observed by their function, for example, a working surface of a desk could be inferred by the objects placed upon it; we call such evidence as the *functional*

evidence. Such a representation can allow us to recognize several instances of desks, not all similar in shape and construction, and not necessarily seen previously, in a robust and efficient manner.

The method is robust because significant surfaces aid in detection of the other significant surfaces. A significant surface can confine a domain of another significant surface. The confined domain allows the system to recover a missed significant surface.

Our current system makes two major assumptions. First, it deals with only planar surfaces. Objects with non-planar surfaces must have sufficient planar surfaces to be recognized. Second, it assumes that an object is in a "standard" pose. A standard pose of an object is the one in which the object is usually found in its natural environment. This standard pose allows the systems to recognize objects efficiently. For example, a desk is expected to be placed with its working surface horizontal.

Currently, our system has models for only two classes of objects: doors and desks. We believe that these are sufficient as landmarks for navigation in many office and laboratory environments. Moreover, we believe that our approach can accommodate a wider range of objects easily.

A block diagram of our system is given in Figure10.2. Note that we do not automatically infer significant surfaces from functional descriptions; this translation is currently done by the programmer.

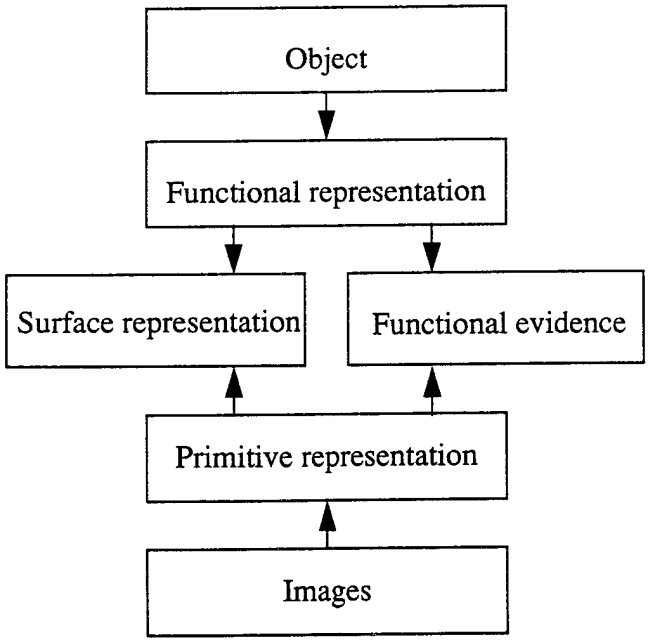


Figure 10.2 Overview of the recognition system

The input to our system is from three cameras arranged to perform trinocular stereo and mounted on a Denning mobile robot. We perform stereo matching on linear line segments detected from the three images. Our matching method is described in detail in [Kim-Neva93] and is similar to one developed at INRIA [Ayac91] but incorporates important junction information. Note that this essentially provides a sparse set of 3-D segments and a somewhat larger set of 2-D segments for the robot to attempt its recognition from.

Detection of the significant surfaces from 2-D and 3-D segments is basically done using perceptual grouping. The detection utilizes the four primitives of the surfaces: orientation, height, shape, and size. The orientation and height are used to reduce search space while the shape and size are utilized for perceptual grouping. Shape of a significant surface may vary for a generic object. Such varying shapes can be detected with help of other significant surfaces. For example, a round desk top surface can be detected with help of legs. The details are described in Section 10.4.

Localization of an object in our system is represented in an *s-map*. The s-map is defined as a map that represents the locations of the visible 3-D surfaces of obstacles in a 2-D space, where 2-D consists of width and length coordinates but does not include a height coordinate. Obstacles are defined as objects that can block the movement of the robot. The s-map is made efficiently from 3-D segments. Further details can be found in [Kim-Neva94].

Section 10.2 explains the significant surface of an object and its primitives. Perceptual grouping for the significant surface is also investigated. Section 10.3 describes recognizing and localizing doors. Section 10.4 resents recognition and localization of desks. Section 10.5 analyzes the recognition system. Finally, Section 10.6 concludes this paper.

10.2 Significant Surface Representation

Significant surfaces follow from the functions that an object performs. Note that one surface may serve several functions whereas a single function may require presence of several surfaces. For example, for a desk, the function of being able to work at a comfortable heights requires a table top within a certain height range as well as some legs for support.

We order the significant surfaces by how essential they are to the functions that they enable. For a desk, we consider the top surface to be more significant than the legs. For an object to be recognized, its most significant surface must be detected.

10.2.1 Primitives of a Significant Surface

A significant surface in a standard pose can be characterized by the four primitives: orientation, range of heights, shape, and size. The orientation and height are decided by the standard pose while the shape and size are determined by the signifi-

cant surface itself. The orientation and height can reduce the number of candidate segments for perceptual grouping to find the surface having the shape and size.

- Orientation: a significant surface of an object has a fixed orientation relative to a horizontal plane in standard pose. For example, a desk top is horizontal.
- Range of height: all the points in a significant surface are within a certain range in terms of their heights above the floor. For example, desk legs have a height range of between zero and 1 meter above the floor.
- Shape: shape of a significant surface may be given in general form. For instance, desk tops generally have a rectangular shape. However, we may be able to detect desk with non-rectangular shape also, based on evidence provided by legs and objects supported on it.
- Size: size of a significant surface of an object is within a certain range. For example, a desk top should have an appropriate size for a human to work on.

10.2.2 Perceptual Grouping to Detect a Significant Surface

Perceptual grouping is used to find a significant surface from 2-D and 3-D segments. The candidate segments can be limited using orientation and height primitives of the significant surface. Perceptual grouping for the significant surface varies depending on its shape. Thus the details of perceptual grouping for each significant surface are explained in the related sections.

In perceptual grouping, 2-D information as well as 3-D information is utilized. This can reduce grouping errors caused by matching errors. The loss of information caused by missing matches and/or partial matches may be recovered using 2-D segment information in two ways. First, missing matches hinder two less meaningful features from becoming a more meaningful one because of lost information. Such lost information can be recovered if there are 2-D segments that can support the meaningful feature. For example, a rectangle with a 3-D U-shape can be classified as a rectangle with higher confidence if it has 2-D segments that can make the U-shape a rectangular shape. Second, partial matches may prevent two less meaningful features from being grouped into a more meaningful feature when proximity of two features is used as one criterion. The partial matches may lose adjacent portion of two less meaningful features, and prevent them from being a more meaningful feature. This error can be overcome if 2-D segments in addition to 3-D segments are used in checking the proximity of two features.

10.3 Recognition of Doors

A door has some functions. The most significant function is for a human and other objects such as furniture to pass through. This decides the most significant surface, a door frame. Another significant function can be separation of space when the door is not used as a passage. This determines another significant surface, a door panel. In addition to these significant surfaces, a door may be supported by functional evidence.

The functional evidence consists of objects seen through a door when it is open. This information helps to decide that the detected door is not a simple drawing but a real door.

First, detecting a door frame as the most significant surface is described. Then detecting a door panel as another significant surface is explained. Next, detecting functional evidence is explored. Finally, the results of detecting doors are illustrated.

10.3.1 Detecting a Door Frame

Adoor frame can be characterized by the following four primitives: a vertical orientation, a height range between a floor and 2.5 meters above the floor, a rectangular shape, and passable size. The orientation and height reduce search space for candidate segments. The shape and size decide a perceptual grouping method to detect the door frame from the candidate segments.

A door frame has three components: top bar, left pole, and right pole. A door frame can be detected by finding a U-shape consisting of the top bar, the left pole, and the right pole.

Candidate segments for top bars, left poles, and right poles are searched in a limited space as described below. The candidate segments are then grouped into door frames. The details of collecting candidate segments and perceptual grouping are described below. In addition, the algorithm to detect door is summarized in Figure 10.3.

```
1. Collect possible top bars
2. Collect left and right poles
3. Hyphothesize doors
   If there is a top bar
     then hypothesize a door with a top bar, left pole, and right pole
     else hypothesize a door with a left pole and right pole
          if there is a 2-D top bar between them
4. Verify a door
   3-D validation with distance and alignment
   2-D validation with distance
```

Figure 10.3 Algorithm for detecting a door frame

Candidate segments are efficiently collected using orientation and size primitives for the three components described above. The orientation and size confine the search space of the candidate segments to vertical surfaces whose height is between a floor and 2.5 meters above the floor. A top bar should be a horizontal line of sufficient length and height, so that a human can pass under the top bar. Poles should be vertical lines that are high enough for a human to pass, and the distance between the two poles should be wide enough so that a human can pass. First, collecting possible top bars are explained. Collecting possible poles are then explored.

Possible top bars are collected from matched segments. A segment can be a possible top bar if it has sufficient height and width. While checking the length of a

matched segment, shrunk 3-D segments due to partial matching can be recovered by considering what portion of 2-D segments are used in reconstructing the 3-D segment.

Left and right poles have the same characteristics in terms of primitives. The poles are also collected among matched segments. A vertical segment can be a possible pole if it is high enough.

Now top bars and poles are perceptual grouped into U-shapes. If top bars of all doors are assumed to be detected, then hypothesizing doors is relatively simple. However, the assumption is not always true. Thus missing a top bar should be considered when hypothesizing doors.

When a top bar is available, a door is hypothesized with a left pole, a top bar, and a right pole. The right pole is in the scope of the top bar. The right pole in the scope of a top bar is a pole below the top bar in an image.

When a top bar is unavailable, a door is hypothesized only with a left pole and a right pole. The right pole is next pole to the left pole. In addition, the distance between them is sufficient for a door. The poles should have a 2-D top bar bridging them. If so, a door is hypothesized with the top bar and two poles.

After hypothesizing a door frame, the door frame is verified with its 2-D and 3-D information. As a 3-D validation, distance and alignment are verified. The distance gap between an end point of a top bar and each pole should be within a threshold value. Moreover, Three components of a door frame should be aligned to a single line in an s-map because they are aligned to a single line at the top view. As a 2-D validation, the distance gap between each end point of a top bar and an upper end point of each pole should be within a threshold value. These verification criteria are also used in selecting a door frame among those hypothesized door frames sharing the same top bar, which can be generated when a door frame has more than two distinct poles. For example, a door frame having two door panels can have more than two distinct poles if a center pole is detected.

10.3.2 Detecting a Door Panel

A door can have several panels. When a door is closed, it is difficult to distinguish panels from a door frame. However, detecting a door panel is easier when a door is open. The panel is attached to a door frame and has a rectangular shape. In the current implementation, our system tries to detect a door panel only when a door is open. The opening of a door is decided using functional evidence described in Section 10.3.3. Detecting a door panel helps to detect a door.

Detecting a door panel is similar to detecting a door frame except that the door panel should be attached to the door frame. Searching for a door panel is done near two poles of the door. After finding an horizontal segment reaching a corner of the door frame, a vertical line reaching the horizontal segment is found. With this horizontal

line and the vertical line, a door panel is hypothesized. The same verification used in door frame verification is applied to verify a door panel.

10.3.3 Finding Functional Evidence

When a door is open to pass, its opening gives functional evidence that constitutes objects seen through the opening. Thus detecting objects inside a door helps to detect the door. To acquire this functional evidence, we collect segments inside a door frame. Then the segments are checked if they are behind the door by a minimum distance set by expected accuracy of depth determination from the viewpoint of the robot. If so, functional evidence for a door is claimed to be achieved. Moreover, these segments behind the door mean that the door is open.

When a door is closed, the functional evidence is not available.

10.3.4 Localization of a Door

After a door is found, the location of the door is represented in an s-map for navigation. Representing the door in an s-map is very simple. A vertical line becomes a point in an s-map because dropping height information of the vertical line renders a point. Thus the door in the s-map is a line linking two points generated from two vertical poles.

In addition to the location of a door, the facing direction of a door should be known to the robot so that the robot is able to reach in front of the door. The facing direction is decided considering locations of both the door and the robot. At first, two locations, which are perpendicular and a predefined distance away from a door, are computed. Then one between the two possible locations is selected based on the distance between the robot and one position. The nearer location is selected because seeing an object means that the front part is always nearer than the back part.

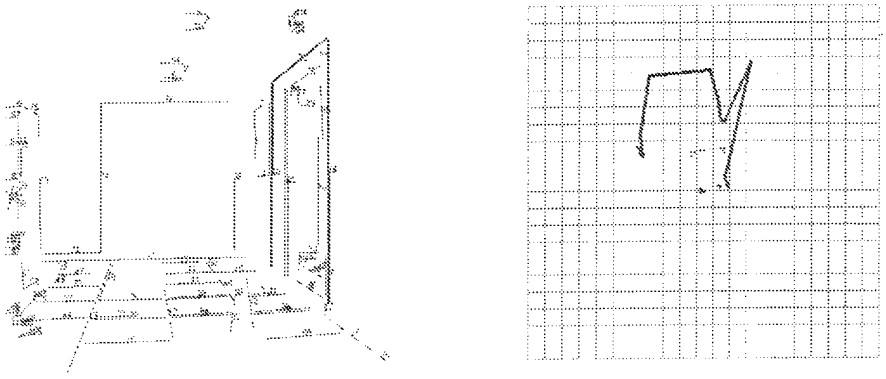
10.3.5 Results for recognition of Doors

Figure 10.4-a and 10.4-b illustrate recognition of an open door for an image scene shown Figure 10.1. In this scene, there are two doors. The right door is open because it has functional evidence of an open door as described in Section 10.3.3. The thicker lines in Figure 10.4-a represents an open door. Moreover, the center position of a door and the front part of a door are represented with small circles bridging the thin line in Figure 10.4-b

The closed door in Figure 10.1 is also detected, but is not shown here separately.

10.4 Recognition of Desks

A desk has the following significant surfaces: the desk top and the legs. The most significant surface of a desk is the desk top. Less significant surfaces are the legs. In addition, evidence of a desk may also be found by detecting the function it performs, namely of supporting objects on the desk top.



(a) Matches (b) S-map

Figure 10.4 Open door

10.4.1 Detecting a Desk Top Surface

The most significant surface for a desk is the top surface. The primitives for a desk top surface are as follows: horizontal orientation, height range between 60 cm and 90 cm, rectangular shape, and workable size. The size is assumed to be from 40 cm to 2 meters in each side. The shape primitive is loosely preserved. Although the system tries to detect the rectangular shape, it allows the desk top to be an arbitrary shape. The desk top is initially detected using perceptual grouping. Then other significant surfaces are used to detect the desk top correctly.

The detection of a desk top using perceptual grouping can be done in two stages: collecting candidate segments for a desk top and perceptual grouping for a desk top. In collecting candidate segments, orientation and height primitives are used to reduce the search space. Segments that are both horizontal and 60-90 cm high above the floor, are collected.

In perceptual grouping, collected candidate segments are grouped into a rectangular shape having workable size. Perceptual grouping is done in four steps: collinearization, L-shapes, U-shape, and rectangular shape. In the first step, possible desk top segments are collinearized based on the angle difference and the gap between two segments. The gap can be as large as the size of a desk because a large part of a desk may be occluded by materials on the desk. In the second step, collinearized lines form L-shapes based on angle and gap between two lines. The angle between two lines should be perpendicular in 3-D. The gap between two lines should be within a threshold value. In checking for a gap, a 2-D gap as well as a 3-D gap is also used to recover errors caused by partial or wrong matches. In the third step, L-shapes form U-shapes. Two L-shape can make a U-shape if they have a common line and the other lines have the same direction. Moreover, the size of a U-shape should be large enough to be a desk. In the final step, U-shapes make rectangular shapes. Two U-shapes can

make a rectangular shape if they have two common lines. The rectangular shape should be large enough for a human to work on.

The detection of a desk top with help of other significant surfaces is done while detecting the other significant surfaces. The other significant surfaces can confine the domain of the desk top. The details are described in Section 10.4.2.

10.4.2 Detecting Legs

Detecting legs of a desk either adds confidence to the desk recognized with a desk top surface, or can help to detect a desk if a desk top surface is not detected. Missing a desk top surface occurs when there are no rectangular shapes, U-shapes, and L-shapes that are large enough because of occlusion by material on the top surface, or by material in front of the desk. This missed top surface may be recovered by detecting legs. While detecting legs, a domain of a desk top is acquired. The details are described below.

Legs have four primitives: vertical orientation, height range between a floor to the desk top surface, no common shape, and size that does not exceeding the 2-D desk domain with respect to an s-map. Orientation and height reduce possible candidates for legs. Then only size primitive is applied to group the candidates. The details of detecting legs are described below. In addition, the algorithm of detecting legs are summarized in Figure 10.5.

1. Collect vertical segments
2. Filter those reaching a desk top surface from the collected segments
3. Select those inside the 2-D domain of a desk from the filtered segments.
   (a) Find the 2-D domain of a desk with respect to an s-map
      i. Find the 1-D domain for a column coordinate
         A. Acquire an initial column domain
         B. Collect possible legs inside the initial column domain
         C. Confine a more accurate column domain from the collected possible legs.
         D. Collect more possible legs inside the more accurate column domain
      ii. Find the 2-D domain in an s-map
         A. Collect the segments inside the column domain
         B. Grow a rectangle containing the collected segments if they are within a reasonable range
   (b) Select those inside the 2-D domain

Figure 10.5 Algorithm for detecting legs

Detection of the legs of a desk is done in three modules: collecting vertical segments, filtering the vertical segments that can reach desk top height, and selecting the vertical segments inside a 2-D domain of a desk top. Collecting vertical segments is done by checking verticality of a segment. The second module is easily implemented by checking 3-D information of vertical segments. The final module needs to find the 2-D domain of a desk with respect to an s-map. After finding the 2-D domain of a desk, more possible legs are selected as legs if they are inside the 2-D domain of a desk. We describe a method of finding the 2-D domain of a desk.

Finding the 2-D domain of a desk is done in two steps: finding a column coordinate domain of a desk in an image, and finding the 2-D desk domain in an s-map.

In the first step, a column coordinate domain becomes more accurate when it interacts with legs. The method attempts to acquire an initial column domain. Then the acquired column domain is used to collect possible legs. Next the possible legs confine a more accurate column domain. Finally, the more accurate column domain is used to collect more possible legs. Among these procedures, we describe acquiring an initial column domain and confining a more accurate column domain because collecting legs in a column domain can be done simply by checking column coordinates of a segment.

Initial column domain is acquired either from segments at a desk height or from a detected desk top surface. The initial column domain should contain all the segments at desk height or segments of the desk top surface.

Confining to a more accurate column domain is done by using a presence row. The presence row is a single row indicating if a column coordinate of the row is occupied by a desk. This presence row is constructed by dropping the row coordinate of possible segments of a desk top surface and marking its coordinate as a filled cell. Thus the region occupied by a desk is marked with filled cells. After making the presence row, the extra possible legs are selected among all the possible legs using the presence row. In the presence row, a band of continuously filled cells is considered as a desk if the band is sufficiently wide. Thus possible legs under such a band become extra possible legs. Conversely, the band having extra possible legs can be considered as a desk. Therefore, such band becomes a column domain of a desk.

In the second step, a 2-D desk domain is computed in an s-map represented in terms of width and depth. A segment with desk top height is collected as a possible desk top segment if it is inside a column domain computed at the first step. After collecting possible desk top segments, a 2-D desk domain is grown by attempting to contain the desk top segments if they are within a certain range. The growing of a 2-D desk domain is described below. In constructing a rectangle containing the segments of a desk, computation time is reduced by transforming segments into another coordinate system. The coordinate system allows checking if a segment is contained to be easily performed by checking its row and column coordinates. Among all the segments of a desk top surface, the longest segment is selected as a reference segment. Then row and column coordinates are rotated so that the reference segment is parallel to row coordinate. This reference segment generates a desk rectangle of which one side is made with the reference segment and the other parallel side is made with a small perturbation of the reference segment. Now other segments are also transformed and their coordinates are compared to see if they are inside the desk rectangle. If a segment is not inside the desk rectangle, the desk rectangle is updated so that it can contain the segment unless the segment is too far. Finally, the desk rectangle is inversely transformed to a world coordinate system.

10.4.3 Finding Functional Evidence

The function of a desk, to work on, can generate functional evidence. When some objects are on the desk, these provide functional evidence. Thus objects on the desk can help recognize the desk. The objects on the desk should be inside the 2-D domain of the desk and reach the desk top.

Detecting materials on a desk top is accomplished in two steps. In the first step, segments inside a 2-D desk domain are collected. These segments can be efficiently collected using 1-D and 2-D filtering. For a 1-D filtering, segments inside the column domain of the desk are collected. Then the collected segments are further checked if they are inside the 2-D desk domain in an s-map. In the second step, segments reaching the desk top are selected among segments inside. In the current system, segments one or both of whose ends reach the desk top are considered as segments reaching the desk top.

10.4.4 Localization of a Desk

After a desk is found, localization is done in an s-map. Location of the desk is simply represented in the s-map by dropping height information of the four sides of the top surface.

In addition to location of a desk, the front direction of the desk should be known to a robot so that the robot can reach the desk and do some other work, such as getting a pencil in a drawer. To find the front part of the desk, common posture is utilized. A desk top has a rectangular shape. Moreover, either of longer sides of the desk top is a front part of a desk. In common posture, the front part of a desk faces a direction that is easily accessible. This implies that the front part of a desk is nearer than its rear part. Now detecting a front part becomes detecting the longer side facing a robot. The detecting of the front part is accomplished in two steps. At first, longer sides of a desk are selected. Then the nearer side between the longer sides is selected as the front part. This selection can be further verified when a robot approaches the desk and acquires more details of the desk front.

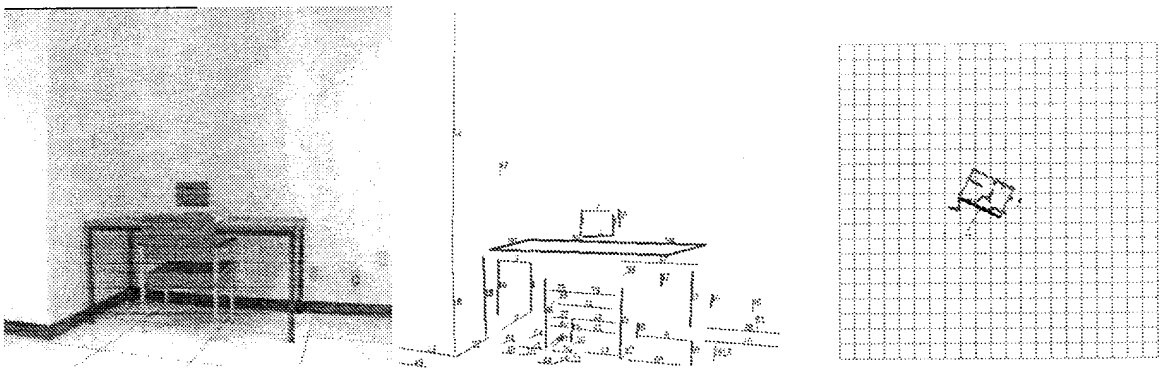
10.4.5 Results for Recognition of Desks

Several tens of experiments have been conducted to recognize four different kinds of desks in many different viewpoints, distances, and settings. Three of the four desks have a rectangular desk top, but have different shaped legs. One of them has drawers in both sides. Another has drawers in one side. The other has no drawers. The desk having a round desk top has only a single leg. The recognition system has successfully recognized the four desks in the experiments with settings of a monitor and a chair. The errors in orientation and size were within 20 percent in experiments done. Results of recognizing desks are given below.

Figure10.6-a shows an image taken by our robot, Antigone. Detecting significant surfaces and functional evidence is given in Figure 10.6-b, -c, and -d. Figure 10.6-b

represents a detected desk top surface on top of matched segments. The thin lines are matched segments. The thicker lines are detected desk top boundaries. The boundaries are generated from a U-shape candidate of a desk top surface because the other side is occluded by a monitor. Figure 10.6-c displays legs under a 2-D desk domain. The current algorithm to detect legs has loose criteria for deciding whether legs are reaching to a desk top because self occlusion and partial matches may prevent legs from reaching to a desk top. Figure 10.6-d illustrates the functional evidence of a desk. In collecting functional evidence, the current algorithm collects only segments reaching a desk top. These significant surfaces and functional evidence allow our recognition system to recognize and localize a desk. The recognized desk is localized in an s-map in Figure 10.6-e. The front part of the desk is represented with the thin line and the small circles.

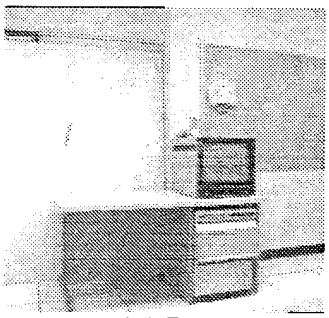
Figure 10.7 displays yet another desk scene. The desk has a monitor on it, and is occluded by a chair. Figure 10.7-b illustrates detected significant surfaces and functional evidence that are represented as thicker lines. All four boundaries of the desk top are successfully recovered although front and rear boundaries of the desk are occluded by a chair and monitor respectively in Figure 10.7-b. Moreover, legs and functional evidence are successfully detected. Figure 10.7-c represents the recognized desk in an s-map. The front part of a desk is rendered with the thin line and the small circles.



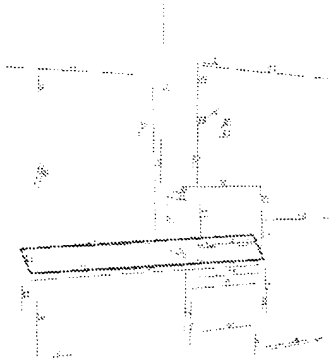
(a) Image (b) matches (c) S-map

Figure 10.7 Desk B

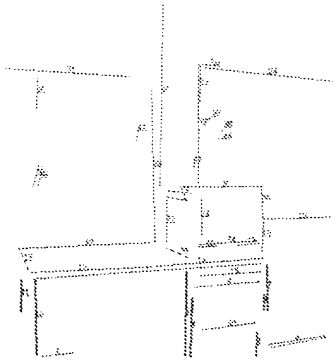
Figure 10.8 displays another desk scene. The desk has a monitor on it. Figure 10.8-b illustrates detected significant surfaces and functional evidence that are represented as thicker lines. All four boundaries of the desk top are successfully recovered although rear boundaries of the desk are occluded by a monitor in Figure 10.8-b. Moreover, legs and functional evidence are successfully detected.



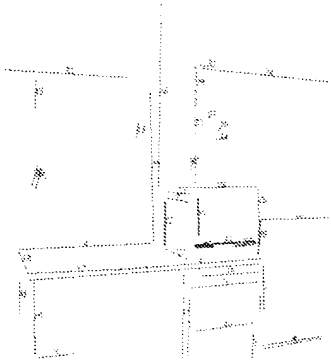
(a) Image



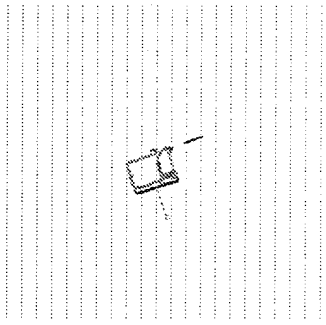
(b) Top



(c) Legs



(d) Functional evidence

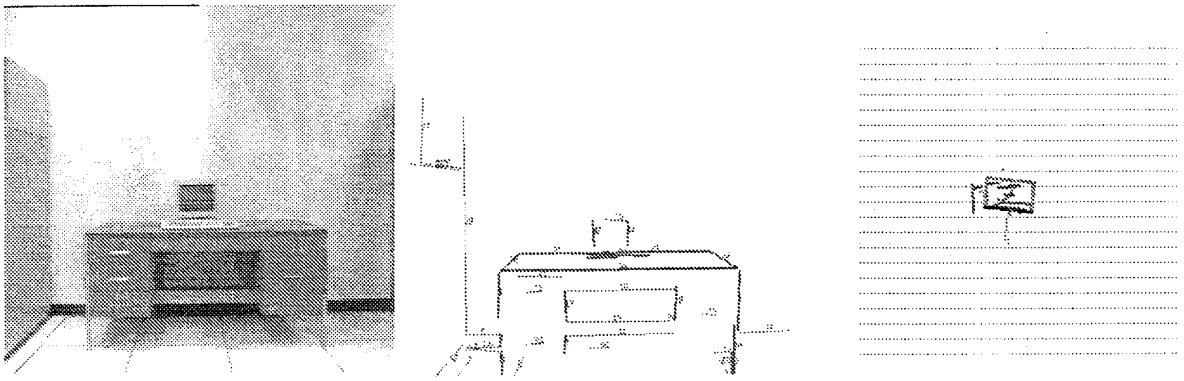


(e) s-map

Figure 10.6 Desk A

Figure 10.8-c represents the recognized desk in an s-map. The front part of a desk is rendered with the thin line and the small circles.

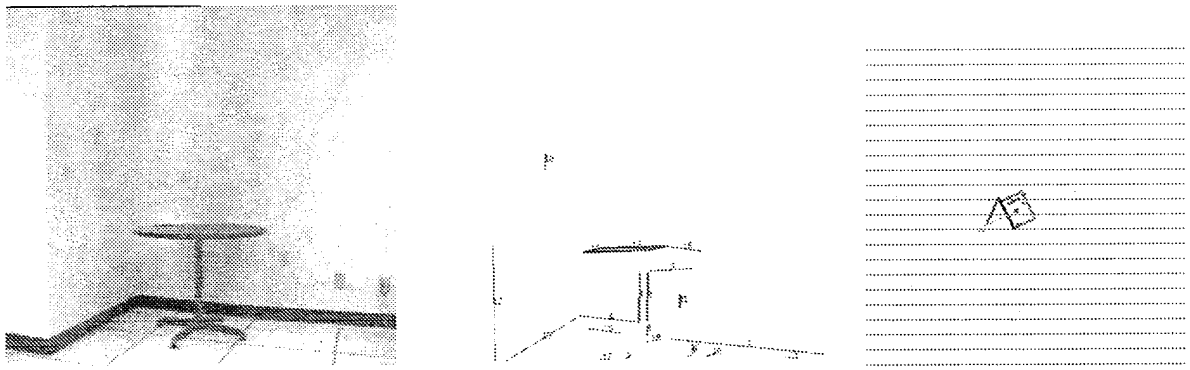
Figure 10.9 displays a round desk scene. Figure 10.9-b illustrates detected significant surfaces that are represented as thicker lines. As seen in Figure 10.9-b, a part of the round desk top is detected by perceptual grouping of a rectangular shape. Then,



(a) Image (b) matches (c) S-map

Figure 10.8 Desk C

the missed part of the round desk top is recovered with help of legs. Figure 10.9-c represents the recognized desk in an s-map. The missed part as well as the detected part is contained in the 2-D desk domain.



(a) Image (b) matches (c) S-map

Figure 10.9 Desk D

10.5 Complexity of the Recognition System

The complexity of the recognition system depends on a target object for recognition. In most cases, the complexity of the recognition system is decided by that of detecting the most significant surface. Because the most significant surface is a key for recognition and helps to confine domains of other significant surfaces, it should be detected correctly or at least have its domain selected roughly.

In the case of recognizing a door, worst case happens when half the segments are vertical lines and half the segments are top bars whose scopes are whole images. Then

$O(n^2)$ hypothesized door frames are generated. However, the complexity of detecting a door frame can be $O(n)$ in the average case because the scope of a top bar reaches two vertical lines in most cases. The complexity of other significant surfaces and functional evidence is $O(n)$. The total complexity for recognizing a door is $O(n)$ in the average case or $O(n^2)$ in the worst case.

In the case of recognizing a desk, worst case happens when all segments are desk top segments. Then generating L-shapes has complexity of $O(n^2)$. Moreover, generating rectangles from L-shapes also has complexity of $O(n^2)$ in the worst case. Therefore, the complexity of detecting a desk top is $O(n^2)$ in the worst case. However, we can reduce candidate segments for a desk top by utilizing natural posture and primitives of the desk top surface. In most cases, the number of the segments of a desk top is less than some constant number. These constant number of the desk top segments allows the system to have the complexity of $O(n)$. The complexities of other significant surface and functional evidence are also $O(n)$. The total complexity for recognizing a door is $O(n)$ in the average case or $O(n^2)$ in the worst case.

We have analyzed the real computing time for recognition from 3-D segments. The computing time was measured in tens of laboratory scenes using Sun Sparc station 10. Although the current system has been written in Lisp without optimization, it showed promising results in terms of computing time. For the case of doors in laboratory scenes, the computing time for recognition was less than one hundredth of a second. For the case of desks in laboratory scenes, the computing time was less than one tenth of a second. In addition to this computation, the edge detection takes about 40 seconds per image. The matching also needs about 10 seconds. From matched segments, an s-map is constructed in less than one tenth of a second. Among these, we estimate that edge detection and matching may be done within a second with parallel processing at a reasonable cost. Thus we believe that total computing time from images to object recognition can be less than a second with low level parallel processing.

10.6 Conclusion

We have shown some experiments on generic object recognition of desks and doors by using representations inspired by their functionality. Some of the evidence we use is rather weak by itself, however, it suffices in the context in which such objects are found. We believe that our methodology can also be applied to other large objects commonly found in offices and laboratories.

10.7 References

[Ayac91]N. Ayache. *Artificial Vision for Mobile Robots*. The MIT Press, 1991.

[Broo83]R. A. Brooks. Model-based three-dimensional interpretations of two-dimensional images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):140–150, 1983.

[Garv-Tene74]T. D. Garvey and J. M. Tenenbaum. On the automatic generation of programs for locating objects in the office scenes. *Second International Joint Conference of Pattern Recognition*, pages 162–168, 1974.

[Grim89]W. E. L. Grimson. On the recognition of parameterized 2d objects. *International Journal of Computer Vision*, 3:353–372, 1989.

[Kim-Neva93]D. Kim and R. Nevatia. Indoor navigation without a specific map. *Intelligent Autonomous Systems*, 1993.

[Lowe87]D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[Marr-Nish77]D. Marr and K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London*, B(200):269–294, 1977.

[Neva-Binf77]R. Nevatia and T. O. Binford. Description and recognition of complex-curved objects. *Artificial Intelligence*, 8:77–98, 1977.

[Star-Bowy91]L. Stark and K. Bowyer. Achieving generalized object recognition through reasoning about association of function to structure. *IEEE Trans. on the Pattern Analysis and Machine Intelligence*, 1991.

[Tene73]J. M. Tenenbaum. On locating objects by their distinguishing features in multisensory images. *Computer Graphics and Image Processing*, 2:308–320, 1973.

11 List of Publications

Y. Chen and G. Medioni, "Surface Level Integration of Multiple Range Images", in *Proceedings of the Workshop on Computer Vision for Space Applications,* Antibes, France, September 1993.

V. Prasanna, C.-L. Wang, and A. Khokhar, "Low Level Vision Processing on Connection Machine CM-5," in *Workshop on Computer Architectures for Machine Perception,* pp. 117-126, 1993.

D. Kim and R. Nevatia, "Indoor Navigation without a Specific Map," in *Proceedings Workshop on Intelligent Systems*, Pittsburgh, PA, 1993

H. Rom and G. Médioni, "Hierarchical decomposition and axial shape description," In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 10, pp. 973-981, October 1993.

H. Rom, *Part Decomposition and Shape Description*, Ph. D. Thesis, University of Southern California, December 1993.

C. Liao and G. Medioni, "Surface Approximation of a Cloud of 3-D Points," CAD94 Workshop, Pittsburgh, PA.

P.Havaldar, G.Médioni and F. Stein, "Extraction of Groups for Recognition," in *Proc. ECCV94*, Stockholm, May 1994.

M. Zerroug and R. Nevatia, "Segmentation and 3-D recovery of SHGCs from a single intensity image," *European Conference on Computer Vision*, Stockholm, May 1994, pp. 319-330.

Y. Chen and G. Medioni, "Shape Description of Complex Objects from Multiple Range Images", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1994, Seattle WA., pp.153-158.

M. Zerroug, *Segmentation and Inference of 3-D Descriptions from an Intensity Image,* Ph. D. Thesis, University of Southern California, June 1994.

Y. Chen, *Description of Complex Objects from Multiple Range Images*, Ph. D. Thesis, University of Southern California, August 1994.

M. Bejanin, A. Huertas, G. Medioni and R. Nevatia, "Model Validation for Change Detection," *Proceedings of the 1994 ARPA Image Understanding Workshop, Monterey,* California, November 1994.

Y. Chen and G. Medioni, "Surface Description of Objects from Multiple Range Images," in *Proc. ARPA Image Understanding Workshop*, Monterey, CA, November 1994.

G. Guy and G. Médioni, "Inferring Surfaces from Sparse 3-D Data," in *Proc. ARPA Image Understanding Workshop*, Monterey, CA, November 1994.

D. Kim and R. Nevatia, "A Method for Recognition and Localization of Generic Objects for Indoor Navigation," in *Proc. ARPA Image Understanding Workshop*, Monterey, CA, November 1994.

C. Liao and G. Médioni, "Surface Approximation of Complex 3-D Objects," in *Proc. ARPA Image Understanding Workshop*, Monterey, CA, November 1994.

N. Milhaud and G. Médioni, "Learning, Recognition and Navigation from a Sequence of Infrared Images," in *Proc. ARPA Image Understanding Workshop*, Monterey, CA, November 1994.

V. Prasanna and C.-L. Wang, "Image Feature Extraction on Connection Machine CM-5," in *Proc. ARPA Image Understanding Workshop*, Monterey, CA, November 1994.

H. Rom and G. Médioni, "Part Decomposition and Description of 3D Shapes," In *Proc. ARPA Image Understanding Workshop*, Monterey, CA, November 1994.

M. Zerroug and R.Nevatia, "Three-Dimensional Part-Based Descriptions from a Real Intensity Image," in *Proc. ARPA Image Understanding Workshop*, Monterey, CA, November 1994.

D. Kim, *Indoor Navigation with a Generic Map*, Ph. D. Thesis, University of Southern California, 1994.

M. Zerroug and R.Nevatia, "From an Intensity Image to 3-D Segmented Descriptions." in *Proceedings of the IEEE International Conference on Pattern Recognition*, Jerusalem 1994.

M. Zerroug and R. Nevatia, "Volumetric descriptions from a single intensity image," to appear in *International Journal of Computer Vision*.

Y.C. Kim and K. Price, "A Feature-Based Monocular Motion Analysis System Guided by Feedback Information," Submitted to *IEEE Trans PAMI*, January 1995.

12 Professional Personnel

12.1 Personnel

Professional personnel included Dr. R. Nevatia, Dr. G. Medioni, Dr. K. Price, A. Huertas, Y. Chen, G. Guy, S. Han, P. Havaldar, D. Kim, M. Lee, C. Liao, H. Rom, M. Zerroug.

12.2 Ph. D. Graduates

In this contract period we have had 4 Ph. D. graduates:

H. Rom, *Part Decomposition and Shape Description*, Ph. D. Thesis, University of Southern California, December 1993.

M. Zerroug, *Segmentation and Inference of 3-D Descriptions from an Intensity Image,* Ph. D. Thesis, University of Southern California, June 1994.

Y. Chen, *Description of Complex Objects from Multiple Range Images*, Ph. D. Thesis, University of Southern California, August 1994.

D. Kim, *Indoor Navigation with a Generic Map*, Ph. D. Thesis, University of Southern California, 1994.